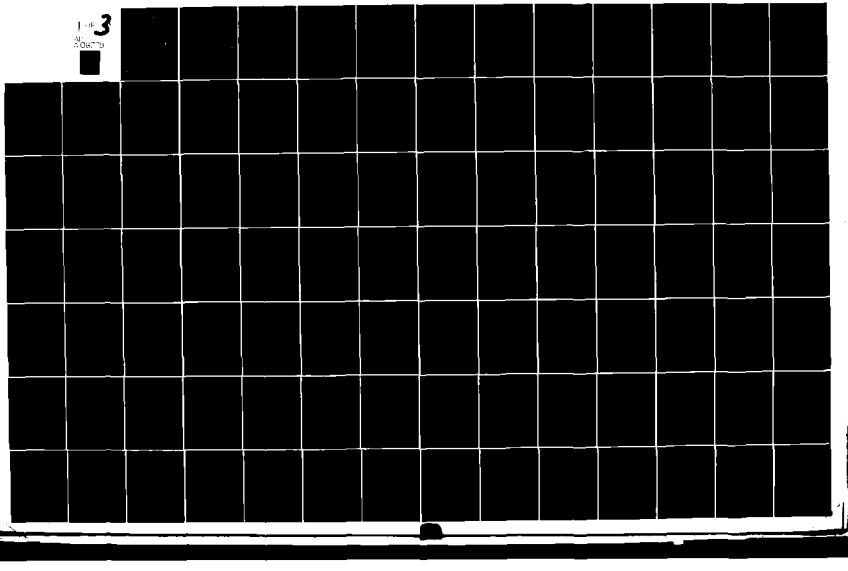
AD-A106 779 FLORIDA INST OF TECH MELBOURNE DEPT OF ELECTRICAL AN--ETC F/G 14/2
IMPLEMENTATION OF THE RECOMMENDATIONS MADE ON THE TECHNICAL REP--ETC(U)
JUN 81 J HADJILOGIOU AFOSR-81-0120

UNCLASSIFIED AFOSR-TR-81-0704 NL



LEVEL II

12

IMPLEMENTATION OF THE RECOMMENDATIONS

MADE ON THE TECHNICAL REPORT TITLED

"ANALYSIS OF ADVANCED SIMULATOR FOR PILOT TRAINING"

FINAL REPORT

TO

Air Force Office of Scientific Research
Bolling Air Force Base, DC 20332

PREPARE UNDER:

Grant [redacted]
AFOSR-81-0120

Submitted By:

John Hadjilogiou, Ph.D., P.E.
Department of Electrical & Computer Engineering
Florida Institute of Technology
Melbourne, Florida 32901
(305) 723-3701

30 June 1981

Approved for public release;
distribution unlimited.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFOSR-TR-81-0704	2. GOVT ACCESSION NO.: AD-A106 779
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): IMPLEMENTATION OF THE RECOMMENDATIONS MADE ON THE TECHNICAL REPORT TITLED 'Analysis [redacted] OF ADVANCED SIMULATOR FOR PILOT TRAINING'	5. TYPE OF REPORT & PERIOD COVERED: Final Report.
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): John Hadjilogiou	8. CONTRACT OR GRANT NUMBER(s): AFOSR-81-0120
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Florida Institute of Technology, Dept of Electrical & Computer Engineering, Melbourne, Florida 32901	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F, 2313/T7
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Office of Scientific Resch/NI, Bolling AFB, DC 20332	12. REPORT DATE: June 1981
	13. NUMBER OF PAGES: 109
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

16, 2313

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microprogrammable processor, control logic for 32175 computer mocro coding of ASPT simulator.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This project resulted in a report detailing specific quidelines for writing and testing custom micro-programs for the 32/75 computer. The micro-program instruction format is analyed in detail and then illustrated by a concrete example.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified 411305 LB

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT

This study which was conducted at Florida Institute of Technology was to implement some of the recommendations made on the technical report titled "Analysis of Advanced Simulator for Pilot Training".

The final report specify guidelines for writing custom micro-programs routine for the 32/75 computer. It includes the hardware feature of the digital system with special emphasis on micro-program control section. The micro-instruction format is analyzed in detail to allow the reader to follow the example of the SEL report reproduced on the Appendices.

FOREWARD

The study was conducted by the faculty of the Electrical and Computer Engineering Department at Florida Institute of Technology.

Dr. John Hadjilogiou Professor and Principle Investigator and Dr. Kofi Torku, Assistant Professor of the Department were the major contributors of this investigation.

The Final Technical Report was typed by Stella Stehno and Marjorie Quaiel.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

1.0 INTRODUCTION

Routines coded in microprogram are similar to those coded in Assembly or high level language in that a coherent sequence of instructions is used to execute various commands required by the computer in both cases. However, speed improvement as high as 16:1 can be obtained when routines are coded in microcode.

A microinstruction usually has two parts:

a) The definition and control of all elemental microoperations to be carried out.

b) The definition and the control of the address of the next microinstruction to be executed.

The elemental microoperations to be carried out are a function of the machine being controlled and, as such, a good knowledge of the data structure and architecture of the machine is required. For the SEL 32/75 the definition of the various micro-operations to be carried out included such things as the ALU, A-MUX, B-MUX, Literal Generator, etc.

1.1 Next Address Control

It is necessary to execute sequences of microinstructions as defined by a microinstruction Sequencer. The Sequencer must provide for:

a) Continuing from one instruction to the next sequential microinstruction. In this mode the Sequencer simply acts as an address counter.

b) Microprogram jumping. This feature allows the Sequencer to select a microinstruction other than the next microinstruction.

c) Conditional jumping. This feature enables a jump to be made to a specified address based on the result of some test.

b) Subroutining in microprogram. This allows a block of microcode (or a single microcode) to be shared by several microinstructions. This requires the sequencer to store the address to which the subroutine should return when it has completed its execution.

To be conversant with any microprogrammed machine and be able to write microcode, one needs to study how the above four functions of a sequencer are implemented.

For example, in the case of the 32/75 computers, the S-field is used to control the address sequencing. Conditional branching is obtained by using the S-field in conjunction with the T-field. We will return to this later.

1.2 The 32/75 Microprogram Timing

Two μ-cycles or machine cycles are generally required to execute a microinstruction. These are called the CROM-cycle and CREG-cycle. Each μ-cycle is 150 nanoseconds long. During the CROM cycle, the basic tests and sequencing are done while all other orders are executed during the CREG cycle. Because of the pipelining nature of how the instructions are fetched and executed, there is an overlap of how the instructions are executed. This is shown in Fig. 1.0 · While one instruction is going through the CREG cycle, the next one is going through its CROM cycle simultaneously. This allows a shorter microcode cycle and speeds up execution of the microprogram.

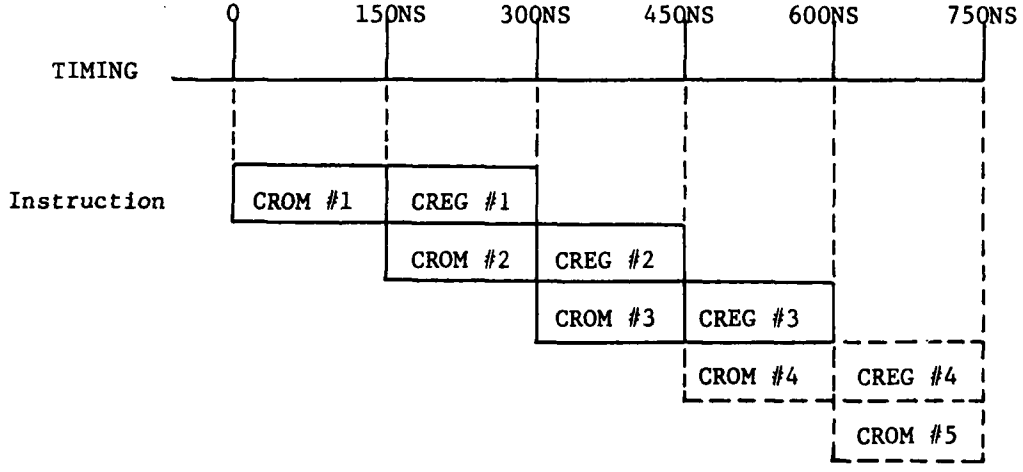


Figure 1.0 (ROM/CREG) Cycle Relationship

2.0 HARDWARE FEATURE OF THE 32/75 COMPUTER

The 32/75 series computer is divided into two sections as far as the hardware is concerned:

a) Microprogrammable Processor (MP)

b) Data Structure (Section)

2.1 Microprogrammable Processor (MP)

The Microprogrammable Processor executes the microprogram to control the 32/75 computer.

The major functional elements of the Microprogrammable Processor (MP), organized around the Control Read-Only Memory (CROM) are:

1. Control Read-Only Memory (CROM)
2. Test Structure
3. Sequence Control
4. CROM Addressing
5. Order Structure
6. Decode ROM(DROM)

A block diagram of the Microprogrammable Processor is given in Fig. 2.1

2.1.1 Control Read-Only Memory (CROM) & Writable Control Store (WCS)

The CROM consists of several Read-Only Memories which are used to store the microprograms used to decode and execute the computer instruction set. The microprograms in CROM also provide general housekeeping functions such as recognizing and vectoring to trap and interrupt servicing routines, and interrupt prioritizing.

The Writable Control Store (WCS) consists of one or two 64x2K RAM boards that are interfaced to the CROM and serve as a user programmable CROM expansion.

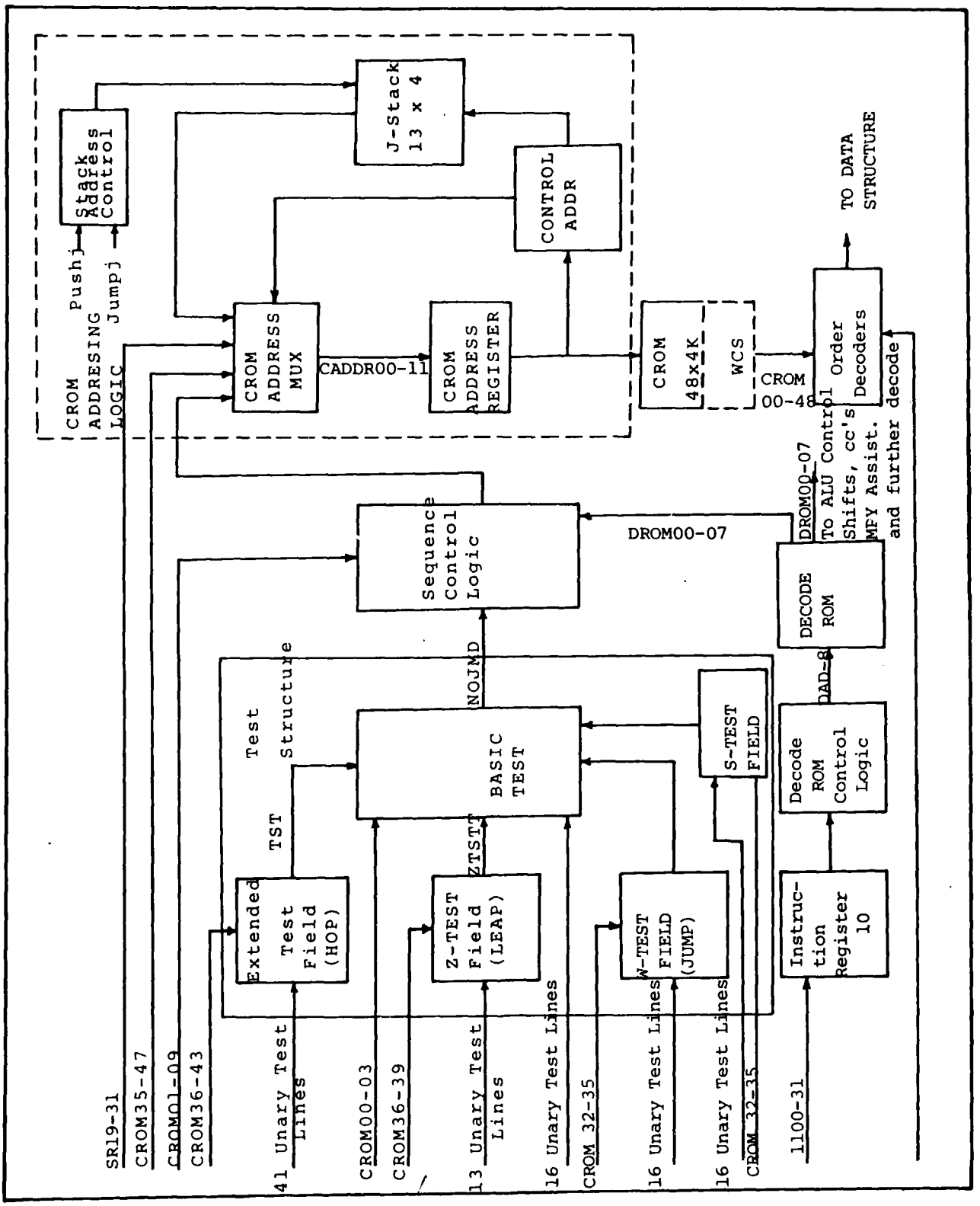


Figure 2-1 Block Diagram - CPU Microprogrammable Processor

Microprograms written for execution in the WCS are virtually the same as those written for the CROM. Hence in the next sections when describing the microprogram, reference will be made only to the CROM.

2.1.2 Test Structure

The basic tests are the first portion of the Micro-Instructions to be executed. There are one basic test field and four secondary test fields associated with the test structure. They are: The Primary Test field, the Extended Test field, the Z-Test field, the W-Test field, and the S-Test field. All Micro-Instruction testing is completed prior to execution of microcode orders.

A block diagram of the test structure is shown in Fig. 2.2

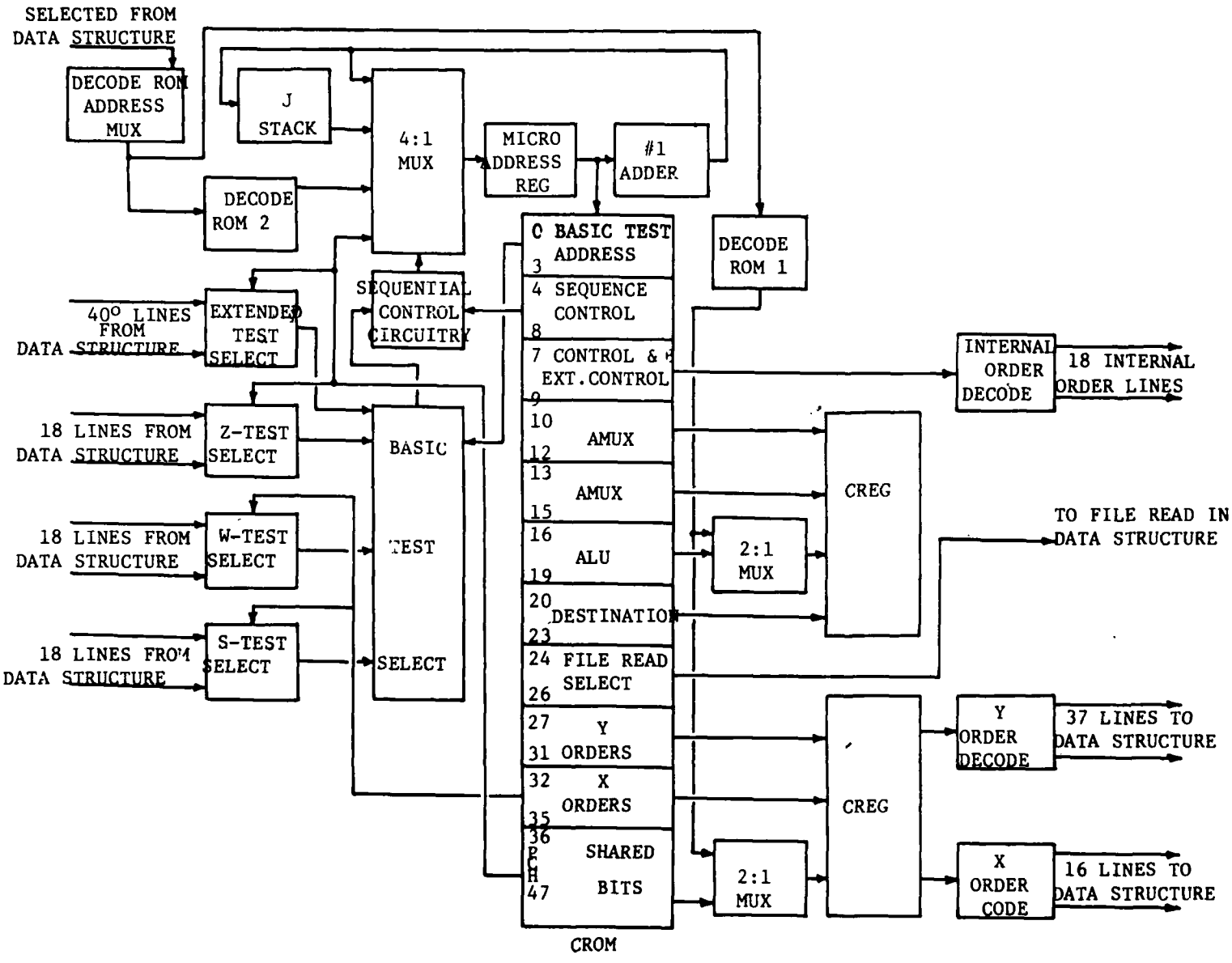


Figure 2.2 32/75 Test Structure

2.1.3 Sequence Control

The Sequence Control logic comprises that portion of the hardware that selects the next CROM address for use by the microcode. This logic is interfaced with the output of the Decode ROM, the Sequence Control logic, and the CROM Address Mux. (Fig. 2.1)

2.1.4 CROM &WCS Addressing

The CROM and WCS Addressing logic provides the capability of addressing the CROM (and WCS) from one of several sources. The logic circuitry consists of the CROM Address Mux, CROM Address Register, a J-Stack and a CROM Address Adder (Fig. 2.1).

1. The CROM Address Mux is a 4:1 multiplexer which selects one of four address sources to formulate the next CROM Address.
2. The CROM Address Mux is influenced by the Sequence Decode circuit. With a Sequence Control field of one (JUMPJ), the next microinstruction address will be conditionally taken from the top of the J-Stack. This effectively is a Return instruction from a branch.
3. The J-Stack is a 4-deep push-down stack with a usable depth of three which is pushed or popped by a jump taken as a result of an instruction execution. A JUMPJ instruction gates the BJOO-12 bits to the inputs of the CROM Address Mux. The JUMPZ with the test met disables the CROM Address Mux forcing the next instruction to be obtained from address 0.

The CROM Address Register holds the CROM Address that is currently being executed.

The CROM Address Adder is used to generate addresses that are applied to the J-Stack and the CROM Address Mux.

2.1.5 Order Structure

The Order Structure contains the logic necessary to decode the CROM and/or CREG bits for the direction and control of the operation to be performed. The Order Structure consists of the following: (Fig. 2.3)

1. Y-Order
2. X-Order
3. U-Order
4. Conditional Order

The various fields of the microinstruction that control the respective orders are discussed in later sections.

2.1.6 Decode ROM

The Decode ROM (DROM) and associated logic (Fig 2.1) are used to enable each Macro-instruction fetched from main memory to vector to the location in CROM microroutine that executes that particular Macro-instruction.

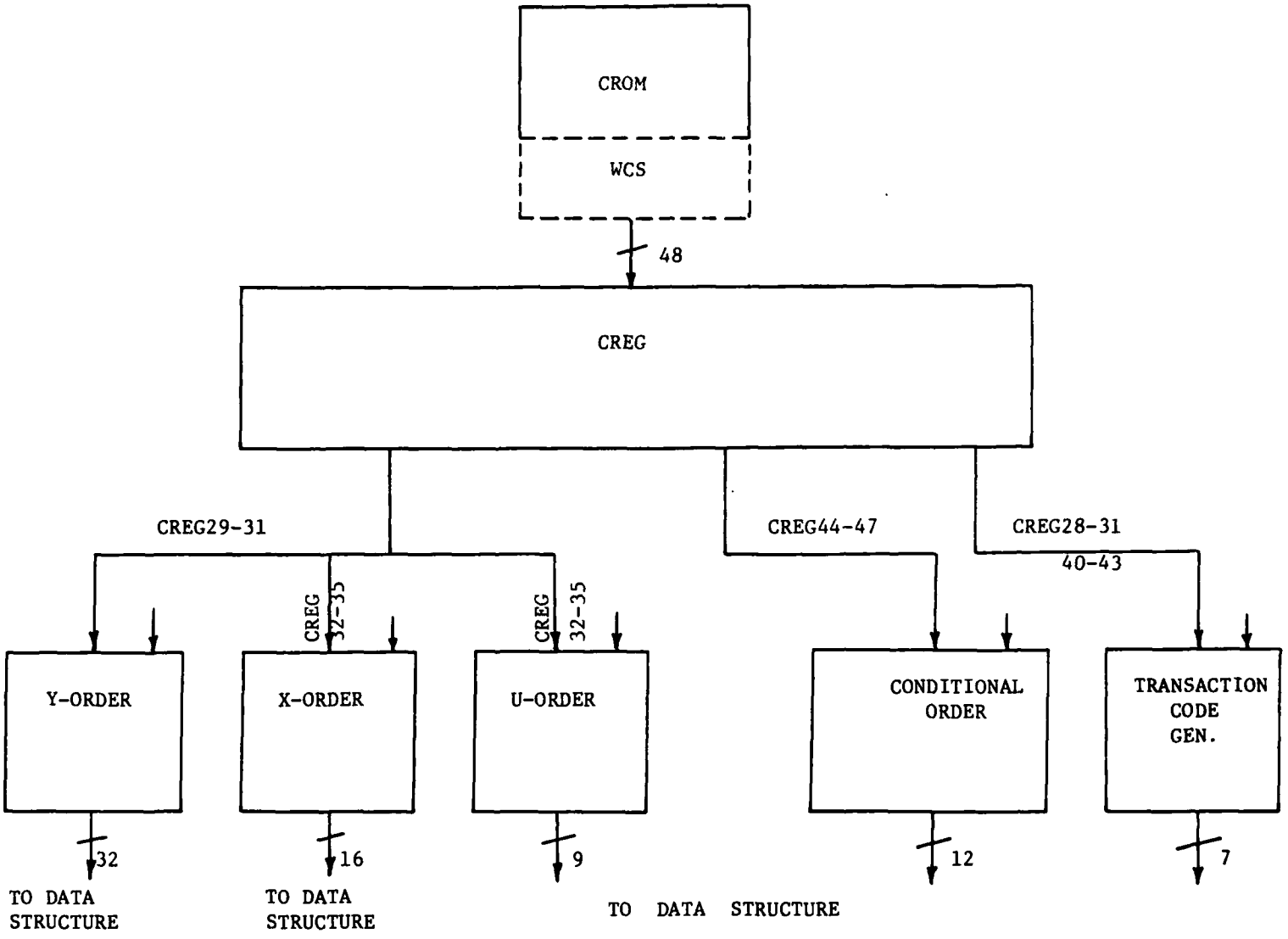


Fig. 2.3 Block Diagram Order Structure

2.2 Data Structure

The Data Structure, shown in Figure 2.4, contains a 32 x 32-bit General File register, hardware registers and two multiplexers organized around an Arithmetic Logic Unit, and a 256 x 32-bit Local Store. The hardware registers are used for SelBUS communications, temporary storage, and shifting. The Data Structure consists essentially of the following:

1. Arithmetic Logic Unit (ALU)
2. A-Multiplexer (AMUX)
3. B-Multiplexer (BMUX)
4. Literal Generator (LIT)
5. General File Registers (FILE)
6. Memory Address Register (MAR)
7. Program Counter Register (PC)
8. N-Counter Register (NCTR)
9. Shift Register (S)
10. Temporary Register (T)
11. Data Input Register (DI)
12. Instruction Decode Register (I0)
13. Instruction Pipeline Register (I1)
14. Local Store (SCRATCH)
15. Bit Mask Generator (BMG)

2.2.1 Arithmetic Logic Unit (ALU)

The ALU is a 2-input, 32-bit Arithmetic and Logical Function Generator utilizing four lookahead carry generators for increased speed of operation. The ALU can generate 15 Arithmetic and Logical Functions for two inputs selected by the A-Mux and the B-Mux. The outputs from the ALU are distributed to the following destination registers:

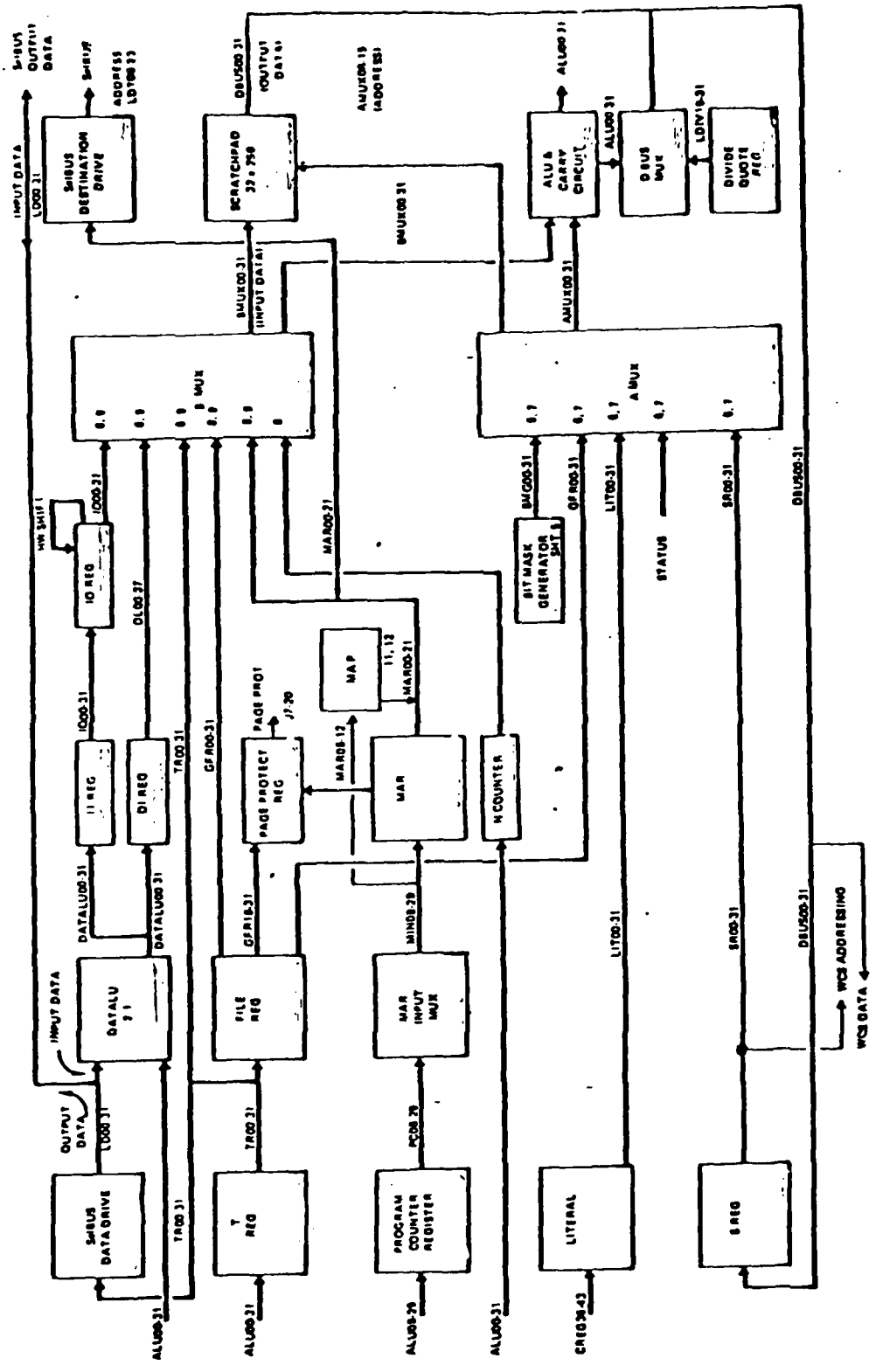


Figure 2.4 Block Diagram - CPU Data Structure

1. General File Register
2. Memory Address Register
3. Program Counter Register
4. N-Counter Register
5. Shift Register
6. Temporary Register (T Reg)
7. Instruction Register 0

The ALU is controlled by the ALU Control Field which chooses the function the ALU is to perform on the input data. The ALU function can also be chosen from the DROM bits ØØ - Ø4. This option holds if the ALU Control Field has the value F in it. These are discussed in Chapter Three.

2.2.2 A-MULTIPLEXER (A-Mux)

The A-Mux selects one of five data sources for input to the ALU. The five A-Mux sources are:

1. General File Register
2. Literal Multiplexer
3. Status
4. Shift Register (S Reg)
5. Bit Mask Generator

2.2.3 B-MULTIPLEXER (B-Mux)

The B-Mux selects one of six data sources for input to the ALU. The six B-Mux sources are:

1. General File Register
2. Temporary Register (T Reg)
3. Data Input Register (DI)
4. Instruction Register 0 (IO)
5. N-Counter
6. Memory Address Register (MAR)

The B-Mux also performs data manipulation functions, such as swapping halfwords and aligning address components.

2.2.4 LITERAL MULTIPLEXER

The Literal Multiplexer forms a 32-bit constant from an 8-bit value in the microprogammable processor. The literal formed is used primarily as a mask or an absolute number.

2.2.5 GENERAL FILE REGISTER

The General File Register is a 32-word by 32-bit file memory. Each register in the file is directly addressable and can either be accessed or

written. The first eight registers are used for the user general purpose registers, R0 through R7; the remaining registers are used by firmware to hold internal information.

2.2.6 MEMORY ADDRESS REGISTER(MAR)

The Memory Address register is a 24-bit register used to address memory or an I/O device by the destination bus. The MAR is used also for temporary storage.

2.2.7 PROGRAM COUNTER REGISTER (PC)

The Program Counter register (PC) is a 22-bit binary counter which contains the virtual address of the most recent instruction fetched from memory. When the Prcgram Counter is used, the formatted address is expanded to a full 24 bits. Incrementing the Program Counter is under the control of firmware; the PC may be incremented by one (for halfword instructions) or by two (for word instructions).

2.2.8 N-COUNTER

The N-Counter is an 8-bit binary up/down counter used by firmware as an iteration counter for repetitive operations, such as shift, multiply, divide, and load and store file.

2.2.9 SHIFT REGISTER (S REG)

The Shift register is a 32-bit temporary register used to perform the following types of shifts:

1. Shift Right - Arithmetic, Logical, and Circular
2. Shift Left - Arithmetic, Logical, and Circular
3. Shift Right - Nibble
4. Shift Left - Nibble

2.2.10 TEMPORARY REGISTER/DATA OUTPUT REGISTER (T REG)

The Temporary register is a 32-bit multi-use register. This register is used to temporarily hold all data to be written into the General File register. On all microinstructions which specify one of the General File registers in the

destination field, the destination data actually goes to the T-register. During the next cycle, the contents of the T-register are automatically written into the appropriate General File register. The T-register is also called Data Output register when it is used to transmit data to memory or I/O devices over the data bus.

2.2.11 DATA INPUT REGISTER (DI)

The Data Input register is a 32-bit register used to receive operands from memory, or data and status from I/O devices. It can also be used as a bit shift register either by itself or coupled with the S-register. When coupled, the DI register is the least significant register.

2.2.12 INSTRUCTION REGISTER 0 (I0)

Instruction register 0 is a 32-bit register which contains the current instruction being executed. The I0 register is also a shift register which swaps the right and left halfwords of the register to provide for halfword instruction execution.

2.2.13 INSTRUCTION REGISTER 1 (I1)

Instruction register 1 is a 32-bit buffer register which receives instructions as they return from memory. The I1 register contains the next instruction to be executed.

3.0 THE 32/75 MICROINSTRUCTION FORMAT

The 32/75 series computer's microinstruction is 48 bits wide (there is an additional 12 bits optional Floating point). These microinstructions are stored in the Control Read Only Memory (CROM). The 48 bits of the micro-instruction are broken down into 11 fields as shown in Fig. 3.1 and 3.2.

3.1 Introduction

The Control Read-Only Memory (CROM) is the control section of the Central Processor Unit (CPU). The CROM contains permanently stored microinstruction words or Elementary Operations (EO) (these two terms are used interchangeably throughout this text). Groups of EOs form microprograms which are read and decoded to generate the signals necessary to control the CPU operations. The EOs are stored in the Micro Control Unit of the CPU (also referred to as the personality board).

This section describes how the EOs control the function of the CPU. The word format of the ROM is described showing the number of bits assigned to each functional field of an EO. This description is followed by a description of the EO Format Chart showing how the chart is sectionalized. A detailed analysis of the EO fields describes how each field is assigned special control functions.

3.2 Microinstruction Word

The individual microinstructions or EOs that are stored in the Control Read-Only Memory (CROM) control the data paths and the execution of CPU functions. The CROM contains 4,096 48-bit words. The microinstruction word format is shown in Figures 3-1 and 3-2.

Each microinstruction contains the following fields:

T - Primary Tests

S - Sequence Control

M - Control/Extended Control

A - A-mux Select

B - B-mux Select

+ - ALU Control

D - Destination

R - File Read Select

Y - Y-Orders

X - X-Orders

U-Orders

S-Orders

W-Tests

S-Tests

PCH - 12-Bit Branch Address

Z-Tests/

8-Bit Branch Address

Extended Test/

4-Bit Branch Address

8-Bit Literal/

Conditional Orders

CC's/

Shift Code/

FPU Order 1

Reg. No./

ROM Page

Flip-Flop 1/

Bus Transfer

Flip-Flop 2

Flip-Flop 3

13-Bit Branch Address

FPU Order 2

Number of Bits	4	3	3	3	3	4	4	3	5	4	12
Field	T	S	M	A	B	+	D	R	Y	X	PCH

Figure 3.1 Microinstruction Fields

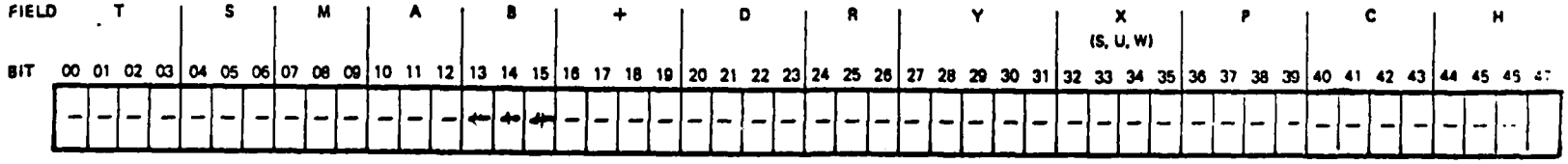


Fig. 3.2 Microinstruction Fields and their Bit Configuration

The following description is an attempt to explain the basic functions of the various fields and how they are related. This should be studied together with the material in the reference (2).

From Fig. 3.2 it can be seen that the microinstruction Bits are numbered from 0 to 47. We will use the notation of the manuals and refer to the Bit Configurations as CROM 00 - CROM 47. The T-field is the first field and occupies CROM 00 - CROM 03 while the last field CROM 44 - CROM 47 belongs to the H-field.

3.3 Primary Test Field (CROM 00 - CROM 03)

This field contains basic test conditions that are used to make decisions during the execution of the microprogram. The entries indicate the type of test being performed. When the test condition pointed to by this field is true or false then a decision can be made by the computer.

If the entry in this field is 0000, no test is selected and is used for unconditional branch by the S-field. This means that the condition selected by the S-field for branch is always true if T=0.

If the entry in this field is 1111, no test is selected. This combination should be used as default value for the T-field.

The following μinstruction fields are influenced by the T-field:

1. The Control Field (M), CROM bits 07-09.
2. The STEST Field (X), CROM bits 32-35.
3. The FPORDER1 Field (H), CROM bits 44-47.
4. The ZTEST Field (P), CROM bits 36-39.
5. The Extended Test Field (PC), CROM bits 36-43.
6. The FPORDER2 Field (P), CROM bits 36-39.
7. The Conditional Order Field (H), CROM bits 44-47.

Both the "true" and "false" conditions are used by the T-field and should be carefully noted. For the case of a "false" test, the condition is true if the test addressed by the T-field is false.

The bit combinations of the T-field that select various tests is explained in Table 3.1.

One point needs clarification in Table 3.1. The S-Test and W-Test use the same fields in microprogram i.e. CROM 32-35. From Table 3.1, when writing ucode, we know that CROM 32-35 will be interpreted as W-test <u>if</u> the T-field has value 3 or 5. On the other hand CROM 35-35 will be interpreted as S-test only if T has the value 4.

Table 3.1 Basic Test (T) Field (CROMOO-03) (2)

Bit 0123	Value	Syntax	Description
0000	0	True	This condition always provides a 'true' test and is used for the unconditional branch.
0000	1	Extended Test True	This condition is met if the test addressed by the extended test field, CROM36-43, is true.
0010	2	Z-Test True	This condition is met if the test addressed by the Z-Test field, CROM36-39, is true.
0011	3	W-Test True	This condition is met if the test addressed by the W-Test field, CROM32-35, is true. Since the X-Order field is used for the W-Test field, X-Orders are inhibited during the CREG cycle of this microword.
0100	4	S-Test True	This condition is met if the test addressed by the S-Test field, CROM32-35 is true. The X-Orders are inhibited during the CREG cycle of this microword. Note: No provision is provided for testing the 'False' condition of the S-Test.
0101	5	W-Test False	This condition is met if the test addressed by the W-Test field (CROM32-35) is false. The X-Orders are inhibited during the CREG cycle of this microword.
0110	6	Z-Test True and Extended Control	This condition is similar to Z-Test True (2); however, it also causes the M-field, CROM07-09 to be interpreted as an Extended Control field.
0111	7	NOEXTUNIV	The No External Universal Condition test is met if both of the following conditions are present: (1) The Enable Interrupt flip-flop (ENAINTFF) is reset, (2) an External Event Global condition is not present.
1000	8	Enable Floating-Point Unit Order, Group 1	This condition is always met and causes the H-field (CROM44-47) to be interpreted as Floating-Point Unit Order, Group 1.
1001	9	Extended Test False	This condition is met if test addressed by the Extended Test field, CROM36-43, is false.

Table 3-1. Basic Test (T) Field (CROM00-03) Cont'd.

	Value	Syntax	Description
1010	A	Z-Test False	This condition is met if the test addressed by the Z-Test field (CROM36-39) is false.
1011	B	ALUZ	The ALU Zero test is met if the ALU output was equal to zero during the CREG cycle of the second preceding microinstruction.
1100	C	NALUZ	The Not ALU Zero test is met if the ALU output was not equal to zero during the CREG cycle of the second preceding microinstruction.
1101	D	Enable Floating-Point Order Groups 1 and 2	This condition is always met and causes the P-field (CROM36-39) to be interpreted as Floating-Point Order Group 2, and the H-field (CROM44-47) to be interpreted as Floating-Point Unit Order Group 1.
1110	E	Z-Test False And Extended Control	This condition is similar to Z-Test False (A); however, it also causes the M-field (CROM07-09) to be interpreted as an Extended Control field.
1111	F	FALSE	This test is never met and is used to inhibit branches, jumps, and conditional orders. The FALSE function is frequently used with *JUMPZ to allow the unconditional logic initialization provided by *JUMPZ, while inhibiting the actual jump to location zero. Note: If the T-field (CROM00-03) is equal to '7', the M-field (CROM07-09) is interpreted as a Floating-Point Unit Control field (vectored jump control field); the P-field (CROM36-39) is interpreted as Floating-Point Unit Order Group 2; and the H-field (CROM44-47) is interpreted as Floating-Point Unit Order Group 1.

3.4 Sequence Control Field (CROM04-06)

The Sequence Control Field (S-field) is used to effect branching in microcode. Each branch that can be taken is considered conditional depending on the results of the test selected by the T-field, Fig. 3,1. This arrangement when used with the T-field value of 0 provides for unconditional branch in microcode. The branch address is external to the microinstruction being executed if the value of the S-field is less than or equal to 3.

If the S-field has value 0 then no branch is taken and this value should always be used as the default value.

The bit combinations of the S-field that select various branch modes are summarized in Table 3.2.

Field	T				S		
Bit	00	01	02	03	04	05	06
	Primary Tests				Sequence		
0	True				0	NOP	
1	Extended Test True				1	JUMPJ	
2	Z-Test True				2	JUMPB	
3	W-Test True				3	JUMPZ	
4	S-Test				4	HOP 4-Bit	
5	W-Test False				4	LEAP 8-Bit	
6	Z-Test True & Extd. Control				6	Branch 12-Bit	
7	No Ext Univ				7	JWCS 13-Bit	
					Branch Code		
8	Enable FPU Group 1						
9	Extended Test False						
A	Z-Test False						
B	ALUZ						
C	NALUZ						
D	Enable FPU Groups						
E	Z-Test False & Extd. Control						
F	Not Used						
Selected Condition for Conditional Branch							

Fig. 3.3 Conditional Branch Codes of Microinstruction

Table 3-2 Sequence Control (S) Field (CROM04-06)

Influenced by:	Results of the Basic Test field
Influences:	1. Conditional Order field (CROM44-47) 2. The address length interpretation of CROM35-47
General:	Each sequencing order which can cause a jump is considered conditional on the result of the test selected by the Basic Test (T) field (CROM00-03). S-field values 0-3 specify that the source of the jump target address is external to the microword (J-stack, Decode ROM, and etc.). Since these types of jumps require no additional resources from the microword, the H-field is interpreted as a Conditional Order field. S-field values 4-7 specify that the source address for the jump is the X-, P-, C-, and H-fields (CROM35-37). Thus, the Conditional Order interpretation of the H-field is inhibited and the H-field is interpreted as the least significant four bits of the jump address. The remaining bits of the X-, P-, and C-fields may have simultaneous multiple interpretations as long as no specific bit conflicts exist. For example, the P-field is used to supply the four most significant bits of a 12-bit jump address and, at the same time, may provide a 4-bit file address, as long as the file address bits exactly match the four most significant bits of the 12-bit branch address.

Bit 654	Value	Syntax	Description
000	0	No operation	This value implies that no jump will occur, regardless of the specified test condition status. The Conditional Order field (CROM44-47) is the only part of the microword which uses the test results.
001	1	*JUMPJ	The Jump based on the J-Stack function implies that the next micro-instruction address is conditionally taken from the top of the J-stack (last address stored in the J-stack). This function provides a micro-instruction equivalent RETURN from a Branch-and-Link. The J-stack is a pushdown stack with a depth of 3, and is only pushed (*LINK function) or popped ("JUMPJ function) by the 'Test True' result of micro-instruction execution. The *JUMPJ function provides a 13-bit jump address so that the target jump address may be in main CROM or Writable Control Storage (WCS).

Table 3-2 Sequence Control (S) Field (CROM04-06) Cont'd.:

Bit 654	Value	Syntax	Description
010	2	*JUMPD	The Jump based on the Decode ROM value implies that the next micro-instruction address is conditionally taken from bits 08-20 of the Decode ROM (D-ROM). This function provides a 13-bit jump address which may point to main CROM or WCS. The D-ROM supplies the jump address for the macro instruction decode process.
011	3	*JUMPZ	The Jump to CROM Location Zero condition implies that the next micro-instruction address is, conditionally, location zero. Since the 13-bit CROM address is forced to zero, the *JUMPZ function may be used to jump to location zero from main CROM or WCS. The *JUMPZ function is the basic exit path of each macro-instruction and, as such, performs some additional implied functions to clean up the microengine in preparation for the execution of the next macro-instruction. Some of the clean-up functions are performed unconditionally (regardless of the status of the specified test) and some are performed conditionally (specified test must be true). The following list describes the unconditional and conditional functions performed at the time that they occur, relative to the *JUMPZ micro-instruction. UNCONDITIONAL AND CONDITIONAL JUMPZ FUNCTIONS: CROM CYCLE Only conditional functions are performed at this time. CREG CYCLE 1. Clear Left Shift Overflow flip-flop 2. Shift right hand flag history registers 3. Clear page select register (force D-ROM decode of 11 register bits 00-05) CREG CYCLE+ 30 ns 1. Set Enable Zero Detect flip-flop 2. Clear Multiply Previous flip-flop 3. Clear S-register

Table 3-2 Sequence Control (S) Field (CROM04-06) Cont'd:

	Value	Syntax	Description
100	4	*HOP *GO TO (4-bit)	This branch 4-bit value causes the four least significant bits of a conditional branch address to be taken from the H-field (CROM44-47). The nine most significant bits of the 13-bit branch address are derived from the CROM address register after the contents of the CROM address register have been incremented by one. The Branch 4 function provides for a branch within a 16- location absolute range.
101	5	*LEAP *GO TO (8-bit)	This branch 8-bit value causes the eight least significant bits of a conditional branch address to be taken from the C- and H-fields (CROM40-47). The five most significant bits of the 13-bit branch address are derived from the CROM address register after the contents of the CROM address register have been incremented by one. The Branch 8 function provides for a branch within a 256-location absolute range.
110	6	*BRANCH *GO TO	This branch 12-bit value causes the 12 least significant bits of a conditional branch address to be taken from P- , C- , and H-fields (CROM36-47). The most significant bit of the 13-bit branch address is derived from the CROM address register after the contents of the CROM address register have been incremented by one. The Branch 12 function provides for a branch within a 4096-location absolute range.
111	7	*JWCS *GO TO (13-bit)	This 13-bit value causes the 13 bits of a conditional branch address to be taken from the X- , P- , C- , and H-fields (CROM35-47). The Branch 13 function provides for a branch within an 8192-location range and may be used to branch from main CROM to WCS or from WCS to main CROM.

3.5 Control Field (M-Field) CROM (07-09)

The M-Field, CROM bits 07-09 is used for two purposes:

Control Field

Extended Control Field

These fields are used in controlling various operations in the data structure of the 32/75 computer.

If the value of the T-Field equals 06H or 0EH then the M-field is interpreted as extended control and the meaning of the various bit combinations is given in Table 3.4. Otherwise, the M-field is interpreted as Control Field and the various bit combinations have the meaning given in Table 3.3.

Table 3.3 Control Field (M-Field) CROM07-09)

Influenced by: Results of the Basic Test Field

Influences: None

General: The following Control field interpretation of the M-field (CROM07-09) only exists if the value of the Basic Test (T) field does not equal'6' or 'E'. If the T-field does not equal '6' or 'E', the M-field is interpreted as an Extended Control field.

Bit 987	Value	Syntax	Description
000	0	No Operation	This value inhibits the Control field.
001	1	SETCC(#)	The Overlay and Set CC's from the D-ROM value causes the D-ROM bits 03-07 to overlay CREG36-39 in order to define the rules by which Condition Codes (CC's) are set. The SETCC (#) function provides the capability for the macro instruction being executed to define the rules for setting CC's. (See Table 3-23)
010	2	SHIFTS(#) SHIFTDI(#) SHIFTD(#)	The Overlay Shift Control and Decrement N-Counter value in the Control field causes D-ROM (bits 04-07) to overlay the Shift Control field (CREG40-43). The Overlay Shift and Decrement N-Counter function provides the macro instruction being executed with the capability of defining the shift type (arithmetic or logical) and the shift direction (left or right). An additional capability is provided to automatically decrement the N-Counter which should contain the shift iteration count. The Overlay Shift and Decrement N-Counter may be used with a shift S-register (SHIFTS(#)); a Shift DI register (SHIFTDI(#)); or a shift double-precision of the S-register and the DI register (SHIFTD(#)). (See Table 3-24.)
011	3	DECRN	This value causes the N-Counter to be decremented at the end of the CROM cysle of the microinstruction.
100	4	DECODE(X)	The Load Lower Decode value causes the page select PROM, at the location specified by 'X', to be loaded into the page select register. The value 'X' must be in the hexadecimal range of '0' to 'F', and stored in bits 40-43 (PROM Page field) of the microinstruction. For this value the most significant bit of the 5-bit page select PROM address is forced to a logical Zero.

101	5	DECODE(1X)	The Load Upper Decode value causes the page select PROM, at the location specified by '1X', to be loaded into the page select register. The value '1X' must be in the hexadecimal range of '10' to '1F', and the value 'X' must be stored in bits 40-43 (PROM Page field). For this value, the most significant bit of the page sleect PROM address is forced to a logical One.
110	6	DECODE(#)	The Load Decode value causes the page select register to be loaded from bits 00-07 of the D-ROM. The DECODE(#) value provedes the capability of specifying additional sublevels of decode and CC's, Shifts, and Arithmetic Logic Unit (ALU) overlays for the macro instruction being executed.
111	7	PUSHJ *LINK	The Push the J-Stack value causes the current CROM address plus one to be pushed into the top level of the J-stack if a branch taken condition is established. If no branch or jump is specified by the S-field, or if the test specified by the T-field is false, the J-stack is <u>not</u> pushed.

Table 3-4. Extended Control Field (M-Field) (CROM07-09)

Influenced by:	Results of the Basic Test field
Influences:	1. The File Read Select field (CROM24-26)
General:	2. The Sequence Control field (CROM04-06)
	The following Extended Control field interpretation of the M-field (CROM07-09) only exists if the value of the Basic Test (T) field is equal to '6' or 'E'. If the T-field does not equal '6' or 'E', the M-field is interpreted as the Control field.

Value	Syntax	Description
0	REGSEL FR(X)	The Register Select value causes the three least significant bits of the value 'X' to be stored in the File Read field (CROM24-26) and to be used as the three least significant bits of the file address. The most significant bit of the 4-bit file address is forced to a logical One. The value 'X' is used to represent a file address and must be number or a name equated to a number which has a value within the range of '8' to 'F'.
1	*JUMPS	The Jump Based on the S-Register function causes the next micro-instruction address to be conditionally taken from the S-register (bits 19-31). This value provides the capability of generating a CROM Address in the data structure and then transferring this address to the CROM Address Mux. Since this value is in the Extended Control field, only Z-tests may be used for conditional jumps from the S-register. If the specified test is true, 13 bits are transferred from the S-register to the CROM Address Mux, so that the target jump address may be in main CROM or in WCS.
2	MPROM	The Multiply PROM value primarily assists the Firmware Multiply function and causes the File Read Select field (CROM24-26) to be overlayed with the output of the Multiply Assist PROM. Since the Multiply Assist PROM is addressed by the four least significant bits of the T-register and the value of Multiply Previous flip-flop, the file address selected by the MPROM order and the Multiply Assist PROM is coordinated with the four least significant bits of the multiplier. The MPROM order also causes the value of the Multiplier Next bit to be supplied by the MPROM and to be saved in the Multiply Previous flip-flop.

Table 3-4. Extended Control Field (M-Field) (CROM07-09) Cont'd.

Value	Syntax	Description
3	DIVMSW	The Divide Most Significant Word value causes the File Read Bank Select bit to be switched for a single file access if the S-register (bit 00) is equal to a logical Zero.
4	DIVLSW	The Divide Least Significant Word value causes the File Read Bank Select bit to be switched for a single file access if the last bit of the quotient being developed (Divide Register, bit 31) is a logical Zero.
5	REPEAT	The Repeat the Current Micro-instruction value causes the micro-instruction in which it is coded to be repeated, and the N-counter to be decremented until the N-Counter count is equal to Zero. When the N-Counter reaches Zero, normal micro-instruction sequencing is resumed.
6	SCALE	The Floating-Point Scale value is used with the firmware Floating-Point to assist in the alignment of operands for add/subtract Floating-Point macro instructions. This order causes a 4-bit branch where the 4-bit branch address is supplied by the Floating-Point Assist Scale PROMS. Note: The SCALE order should not be coded in WCS.
7	NORM	The Floating-Point Normalize value is used by the firmware Floating-Point to assist the post-normalization of Floating-Point results. This order causes a 4-bit branch where 4-bit branch address is supplied by the Floating-Point Assist Normalize PROM. Note: The NORM order should not be coded WCS.

3.6 A-MUX Select (A) Field (CREG10-CREG12)

The A-Multiplexer (A-MUX) selects one of eight data sources for inputs to the Arithmetic and Logic Unit (ALU). These eight sources are:

1. General File Register (FILE)
2. Literal Generator (LIT)
3. Status
4. Shift Register (S)
5. S-Register shifted Left 1 bit (SLEFT)
6. S-Register Nibble shifted Right (SNIBR)
7. S-Register Nibble shifted Left (SNIBL)
8. Bit Mask Generator (BMG)

These eight sources are selected by the 3-bit A-field of the microinstruction (CREG Bits 10-12).

The various bit combinations that select each input are given in Fig. 3-5.

The A-field is also used as literal select field. When A=7, we have the "short literal" (see details Table 3-5). When the Y field = 2, we have the long literal.

Table 3-5. A-Mux Select (A) Field (CREG10-12)

Influenced by: Y-Order (CROM27-31)

Influences: None

General: The A-Mux Select field chooses the source of the A input to the ALU and the source of addressing for the 256 by 32 Scratchpad. When the Y-Order LONGLIT order is present, the A-Mux Select field provides the fill bit and byte select code for the literal generation to the A input of the ALU.

A-Mux Syntax: The Micro Assembler generates A-Mux Select values when the A-Mux is implied as a resource to the operation to the operation to be performed. In general, two types of A-Mux resources may be specified or implied. The first resource type is using the A-Mux to supply the A input to the ALU. The syntax of the ALU expression is:

DEST=AMUX ALU FUNCTION BMUX

The second resource type is using the A-Mux to supply the Scratchpad address. The syntax of the Scratchpad address is:

DEST=SCRATCH(AMUX) or SCRATCH(AMUX)=BMUX

In the above expressions, *DEST must equal a valid ALU* Destination term as described in the Destination field description. ALU FUNCTION must equal a valid ALU term as described in the ALU field description. The BMUX term is optional; however, if it is present, it must equal a valid BMUX term as described in the B-Mux Select field description. The AMUX term may be either a 32-bit literal or a valid A-Mux Select field term as described in the following discussion.

12,11,10	Value	Syntax	Description
000	0	S	The S-Register value selects the 32-bit S-register as the input to the AMUX.
001	1	SLEFT	The S-Register Left Shifted One Bit value selects the 32-bit S-register left shifted one bit as the input to the AMUX. The vacated bit position (bit 31) is filled as determined by the Shift Control field (CREG40-43). The contents of the S-register are not modified unless the S-register is selected as the destination register of the ALU operation.
010	2	SNIBL	The S-Register Left Shifted One Nibble value selects the 32-bit S-register left shifted four bits as the input to the AMUX. The vacated nibble (bit positions 28-31) is filled from the T-register (bits 00-03). The bits shifted out of the S-register (bits 00-03 are lost. The contents of the S-register are not

Table 3-5. A-Mux Select (A) Field (CREG10-12) Cont'd.:

Value		Syntax	Description
010	2	SNIBL	modified by this function unless the S-register is selected as the destination of the ALU operation
011	3	SNIBR	The S-Register Right Shifted Four Bits value selects the 32-bit S-register right shifted four bits as the input to the AMUX. The vacated nibble (bit positions 00-03) is filled with the S-register sign bit (bit 00). The bits shifted out of the S-register (bits 28-31) are lost. The contents of the S-register are not modified unless the S-register is selected as the destination of the ALU operation. Note: The S-Register source data through this mux is not affected by any other concurrent shift orders (e.g., if S contains 1 and the A-Mux order is 2, with Y-order SHIFTS, the A-Mux will pass 10).
12,11,10 100	4	R(X) FR(Y)	The Selected File Register Output value selects the 32-bit file output as the input to the AMUX. For 'R(X)' term, the file register address is specified by the File Read Select field (CROM24-26) or the Register Number field (CROM36-39). For the 'FR(Y)' term, the Extended Control, Register Select method is used to address the file, and only file registers '8' through 'F' may be used. Note: This AMUX code causes a File read and must not be used in a micro-instruction following a File write micro-instruction.
101	5	STATUS	The CPU Status value selects the CPU PSW1 Status bits as the input to the AMUX. The Status bits are formatted as follows: Bit 00 — Privilege Bit Bits 01-04 — Condition Code Bits 1-4 Bit 05 — Extended Operand Indexing (Extended Addressing) Bit 06 — Last Instruction Executed was in the Right Halfword Bits 07-08 — Not Used Bit 09 — MAP Not Valid (LVALID) Bit 10 — MAP Write Protected Note: Bits 09-10 pertain to the map entry addressed by MAR bits 04-08. Bits 11-15 — Not Used Bits 16-31 — Contents of the Memory Protect Register Addressed by MAR bits 05-08.

Table 3-5. A-Mux Select (A) Field (CREG10-12) Cont'd.:

	Value	Syntax	Description
110	6	BMG	The Bit Mask Generator value selects the 32-bit Bit Mask Generator as the input to the AMUX. If the AMUX term is used without the Y-Order 'BMGMR', the byte select code for the Bit Mask Generator is supplied by the 10 register, bits 14-15. If this AMUX term is used with the Y-Order 'BMGMR', the byte select code for the Bit Mask Generator is supplied by the MAR virtual register C-bits (LFC1V and LFC2V signals). In either case, the bit mask code within the byte is supplied by the 10 register bits 06-08.
111	7	LITERAL FILL BYTE SELECT @FFFFFFYZ	This value selects the Literal Generator as the input to the AMUX. The actual literal value 'YZ' is supplied by the 'P' and 'C' fields (CREG36-43), and must be within the hexadecimal range of '0' to 'FF'. Note that the fill bits for this literal (bits 00-23) must be all Ones and that the literal byte is in byte position 3 (bits 24-31).

3.7 Literal Select Field (CREG10-12)

Two types of literals (constants) are generated using the A-MUX field:

1. Short Literal - A-mux field equal to 7
2. Long Literal - Y-Order field equal to 2

A) Short Literal

With the A-Mux field equal to 7, the literal value generated from the microinstruction is selected. The byte, CREG36-43, is presented to the Literal circuitry right justified (byte 3) with the most significant three bytes forced to Ones.

B) Long Literal

With the Y-Order field equal to 2, a long literal is constructed according to the state of CREG bits 10-12 as follows:

CREG10 is the fill bit (One or Zero) which is forced into all 24-bit positions not directly specified in the microinstruction.

CREG11-12 select the byte position in the word where the bits of CREG36-43 are inserted. The CREG 11 and 12 byte selection is shown below.

Table 3.5.B Long Literal-CREG Bits 11 and 12

CREG Bits 11	CREG Bits 12	Byte Selected
0	0	Byte 1
0	1	Byte 1
1	0	Byte 2
1	1	Byte 3

A summary is provided as follows:

When the "LONGLIT" order is used, the Literal Select field, CREG bits 10-12, is interpreted as the fill bit and byte select code for the literal generator. The most significant bit of field, CREG bits 11-12, are used as the byte select code. The actual byte literal is always placed in the P- and C-fields, CREG bits 36-43.

LITERAL SELECT FIELD SYNTAX

The following Select Field orders do reflect the correct Assembler Syntax.
@XY000000 (A=0)

Long Literal where the "XY" term must be in the hexadecimal range of 00-FF. The fill bits are zero and the byte select code is zero.

@00XY0000 (A=1)

Long Literal where the "XY" term must be in the hexadecimal range of 00-FF. The fill bits are zero and the byte select code is one.

@0000XY00 (A=2)

Long Literal where the "XY" term must be in the hexadecimal range of 00-FF. The fill bits are zero, and the byte select code is two.

@00000XY (A=3)

Long Literal where the "XY" term must be in the hexadecimal range of 00-FF. The fill bits are zero, and the byte select code is three.

@XYFFFFFF (A=4)

Long Literal where the "XY" term must be in the hexadecimal range of 00-FF. The fill bits are one, and the byte select code is zero.

@FFXYFFFF (A=5)

Long Literal where the "XY" term must be in the hexadecimal range of 00-FF. The fill bits are one, and the byte select code is one.

@FFFFXYFF (A=6)

Long Literal where the "XY" term must be in the hexadecimal range of 00-FF. The fill bits are one, and the byte select code is two.

@FFFFFFXY (A=7)

Short Literal where the "XY" term must be in the hexadecimal range of 00-FF. The Micro-Assembler evaluates this literal as a short literal so the Literal Select field is interpreted as the A-mux Select field. However, if the "LONGLIT" Y-order is manually generated, this becomes a long literal with a fill bit of one and a byte select code of three.

Notes

In the example above, the "@" preceding the literal value indicates to the Micro-Assembler that hexadecimal notation is being used. Literals may also be expressed in decimal, binary, or an evaluatable expression.

3.8 B-Multiplexer Select (B) Field (CREG 13-15)

The B-mux selects one of nine data sources for input to the ALU. Some of the nine B-mux sources are:

1. General File Register (FILE)
2. General File Register, Halfword Swapped (FILE, HWS)
3. Temporary Register (T)
4. Data Input Register (DI)
5. Instruction Decode Register (IO)

The various bit combinations that select each source are given in Table 3-6.

Table 3-6. B-Mux Select (B) Field (CREG13-15)

Influenced by: None

Influences: None

General: The B-Mux Select field chooses the source of the B input to the ALU. The ALU function specified does not require a B-Mux term for this field to be valid.

The Micro Assembler default value for this field is '1'. This value selects the 8-bit N-Counter and the 24-bit MAR register as the B-Mux input.

B-Mux Syntax: The Micro Assembler generates the B-Mux Select field values when the B-Mux is specified or implied as a resource to the operation to be performed. In general, two types of B-Mux resources may be implied or specified. The first resource type utilizes the B-Mux to supply the B input to the ALU. The syntax of the ALU expressions are:

1. DEST=AMUX.NAMES ALU FUNCTION BMUX.NAMES;
2. DEST=BMUX.NAMES;

The second resource type occurs when a B-Mux input is required for direct bit tests and the ALU function specified does not require a B input resource. The syntax for this type of expression is:

1. DEST=AMUX.NAMES, BMUX=BMUX.NAMES;

In this expression, DEST must equal a valid ALU Destination term as described in the Destination field description. AMUX.NAMES must equal a valid A-Mux term, as described in the A-Mux Select or the Literal Select field descriptions. ALU FUNCTION must equal a valid ALU term, as described in the ALU field description. The BMUX.NAMES terms must be a valid B-Mux term as described in the following discussion.

Value	Syntax	Description
0	T	The T-Register value selects the 32-bit T-register as the input to the B-Mux.
1	N/ MAR	The N-Counter and Memory Register value select the 8-bit N-Counter and the 24-bit MAR register as the input to the B-Mux. The N-Counter and MAR input port is formatted as follows: Bits 00-07 N-Counter bits 00-07 Bits 08-16 Logical MAR bits 00-08 Bits 17-29 MAR bits 09-21 Bits 30-31 Virtual MAR bits 22-23 (C-bits, LFC1V and LFC2V signals)

Table 3-6. B-Mux Select (B) Field (CREG13-15) Cont'd.:

Value	Syntax	Description
		Note 1. If the U-Order ('RDMAP') is true, B-Mux Input bits 08-16 contain Physical MAR bits 00-08. If MAR was loaded by the 'FULLMAR' destination term or if the MAP is turned OFF, both Logical and Physical MAR values will be equal. If the MAP is turned ON and MAR was loaded by the 'MAR1X' or 'MAR' destination terms, Logical MAR bits 04-08 contain the MAP register. Bits 00-03 of Logical MAR are not used by the mapping algorithm. 2. The DI or T-register data input through this Mux is not affected by any other concurrent shift orders. 3. Virtual MAR and MAR 22-23 should agree unless a SelBUS transaction has occurred that causes these bits to act as F- and C-bits.
2	I0	The I0 Register value selects the 32-bit I0 register as the input to the B-Mux.
3	DI	The DI Register value selects the 32-bit DI register as the input to the B-Mux. If this B-Mux term is used while a memory operand read or an I/O final transfer is in progress, the CPU automatically enters a WAIT state until the memory or I/O DRT is received and the CREG cycle of the current micro-instruction can be completed using the DI register data just received from memory or I/O.
4	R(X)[5] FR(Y)	The Selected File Register Output value selects the 32-bit File output as the input to the B-Mux. For the 'R(X)' term, the file register address is specified by the File Read Select field (CROM24-26) or the Register Number field (CROM36-39). For the 'FR(Y)' term, the Extended Control Register Select method is used to address the File and only File registers '8' through 'F' may be used. (See Table 5-4.) Note: This B-Mux code causes a File Read and must not be used in a micro-instruction following a File Write micro-instruction.
5	R(X,HWS) FR(Y,HWS)	The Select File Register Output, Halfword swapped value selects the 32-bit file output, circular shifted 16 bits, as the input to the B-Mux. File output bits 00-15 are presented to B-Mux 16-31 and File output bits 16-31 are presented to B-Mux bits 00-15. For the 'R(X,HWS)' term, the File register address is

Table 3-6. B-Mux Select (B) Field (CREG13-15) Cont'd.:

Value	Syntax	Description
5 cont'd.	R(X,HWS) FR(Y,HWS)	specified by the File Read Select field (CROM24-26) or the Register Number field (CROM36-39). For the 'FR(Y,HWS)' term, the Extended Control Register Select method is used to address the File and only File registers '8' through 'F' may be used. (See Table 5-4.) Note: This B-Mux code causes a File Read and therefore, must not be used in a micro-instruction following a File Write micro-instruction.
6	INTLVL PNLDATA	The Interrupt Level and Panel Data value selects the Serial Panel data input lines and current interrupt polling level. The input lines to the B-Mux are formatted as follows: Bit 00 — UART bit 03 Bits 01-03 — Not Used Bits 04-07 — UART bits 04-07 Bit 08 — Not Used Bits 09-15 — Interrupt Polling Level Bits 16-31 — Not Used
7	None	Not Used

3.9 ALU Control (+) Field (CROM16-19)

The Arithmetic and Logic Unit performs fifteen arithmetic or logical operations on the A and B input lines. The data on the A and B inputs is selected by the A-Mux and B-Mux fields of the micro-inspection. The various ALU functions selected by each bit combination of the + field are shown in Table 3-7.

Table 3-7 ALU Control (+) Field

Influenced by:	None
Influences:	None
General:	The ALU Control field chooses the function that the ALU is to perform on the A and B input data lines. The actual data on these lines is specified by the A-Mux Select and B-Mux Select fields of the micro-instruction. The micro-instruction order structure provides for special orders that either save ALU status or use previously stored ALU status. ALU status includes carry-out, result equal zero, and ALU output sign bit. These orders provide the capability of executing double-precision (64-bit) arithmetic and logical functions in coordinated micro-instructions. The Micro Assembler defaults the ALU Control field to value '0'. This action causes the B-Mux input to the ALU to be transferred to its output lines. For ALU Syntax see reference (3).

Value	Function Description
0	Transfer B-Mux input to ALU output.
1	Transfer ones complement of the B-Mux input to the ALU output.
2	Logical OR the ones complement of the A-mux input with the B-Mux input and transfer the result to the ALU output.
3	Add the A-Mux to the B-Mux inputs and transfer the result to the ALU output.
4	Subtract one from the A-Mux inputs and transfer the result to the ALU output.
5	Subtract the B-Mux inputs from the A-Mux inputs and transfer the results to the ALU output.
6	Add one to the A-Mux inputs and transfer the result to the ALU output.
7	Logically OR the A-Mux with the ones complement of the B-Mux inputs and transfer the result to the ALU output.

Table 3.7 ALU Control (+) Field (CROM16-19) Cont'd.:

Value	Function Description
8	Logically Exclusive OR the A-Mux inputs with the B-mux inputs and transfer the result to the ALU output.
9	Transfer the ones complement of the A-Mux input to the ALU output.
A	Logically OR the A-Mux inputs with the B-Mux inputs and transfer the result to the ALU output.
B	Logically AND the B-Mux inputs with the ones complement of the A-Mux inputs and transfer the result to the ALU output.
C	Logically AND the A-Mux inputs with the ones complement of the B-Mux inputs and transfer the result to the ALU output.
D	Logically AND the A-Mux inputs with the B-Mux inputs and transfer the result to the ALU outputs.
E	Transfer the A-Mux inputs to the ALU outputs.
F	The Overlay the ALU Control field value causes the output of the D-ROM, bits 00-04, to overlay the ALU Control field. This feature provides the macro instruction being executed with the capability of defining the ALU operation code.

3,10 ALU Destination (D) Field (CROM20-23)

The output destination of the ALU can be distributed to any of the following eight Registers BUS:

1. General File Register (FILE)
2. Memory Address Register (MAR)
3. Program Counter Register (PC)
4. N-Counter Register (NCTR)
5. Temporary Register (T)
6. Instruction Pipeline Register (II)
7. Data Input Register (DI)
8. DBUS (with following destination:)
   a. Shift Register (S)
   b. WCS Output Data

The output destination may not necessarily be specified. The default value of '0' in the D Control Field indicates that no destination is desired.

The various bit combinations that route the ALU output to the desired destination are listed and explained in Table 3-8.

Table 3-8 Destination (D) Field (CROM20-23)

Influenced by:		None
Influences:		None
General:		The Destination field chooses the destination register for the ALU output. When a register is chosen as the destination register, it is strobed at the end of the CREG cycle of the micro-instruction.
Value	**Syntax**	**Description**
0	NOD	The No Destination term ensures that a register strobe is not generated. This is the default value
1	S	The S-register condition causes the contents of the 32-bit D-bus to be strobed into the 32-bit S-register. The D-bus normally contains ALU data unless the Scratchpad or WCS was selected as a resource to the micro-instruction. (See X-Order 6.)

Table 3-8 Destination (D) Field (CROM20-23) Cont'd.:

Value	Syntax	Description
2	PC	The Program Counter condition causes the contents of the ALU output bus, bits 08-29, to be strobed into the 22-bit Program Counter. The Program Counter is used for the macro program address and is aligned to the 32-bit data structure so that addresses generated by the PC are 24-bit memory word addresses. The two significant bits, which would normally align to ALU bits 30 and 31, do not exist since they are not required for word type memory addresses.
3	MAR	The Memory Address Register (19-bit) condition causes the contents of the MIN bus, bits 08-31, to be presented to the Logical MAR. If the Nonextended Indexing CPU mode is used, only bits 13-31 of the MIN bus are strobed into Logical MAR, bit 05-23. If the Extended Indexing mode is used, bits 08-31 of the MIN bus are strobed into MAR bits 00-23. The MIN bus normally contains ALU data unless the 'PCTOMAR' order is present or a 'FETCHPC' order is present. If the Mapped mode is used during the CREG+1 operation, Logical MAR bits 04-08 address the MAP registers. At the end of CREG+1, the contents of the MAP register are strobed into Physical MAR bits 00-08 to generate a 24-bit real address in Physical MAR (00-08) and Logical MAR (09-23). If the MAR is used as a destination, and the output portion of a previously coded SelBUS transfer is not complete, the CPU enters an automatic WAIT condition. This action is necessary to prevent the destruction of the MAR contents which must be valid for the successful completion of the SelBUS transfer.
4	SCRATCH (AMUX.NAMES)=	With the Scratchpad condition, the contents of the 32-bit B-Mux are strobed into the Scratchpad location addressed by the A-Mux (bits 08-15). The ALU functions are disabled during this micro-instruction.
5	I1	With the I1 condition, the 32-bit contents of the DATA/ALU bus are strobed into the I1 register. The DATA/ALU bus normally contains SelBUS data until the 'I1' or 'DI' destination term is programmed. Because of this conflict, the I1 destination must not be used under the following conditions: 1. If a memory operand fetch is in progress 2. If an instruction fetch is in progress 3. If a Read Status Transfer (RSTX) is in progress 4. If an Interrupt Control Transfer (ICT) is in progress. The I1 destination term sets the CPU I1 full condition and clears any pending I1 conditions that may be present.

Table 3-8. Destination (D) Field (CROM20-23) Cont'd.:

Value	Syntax	Description
6	DI	The DI register condition causes the contents of the 32-bit DATA/ALU bus to be strobed into the DI register. The DATA/ALU bus normally contains SelBUS data until the 'I1' or 'DI' destination term is programmed. Because of this conflict, the 'DI' destination must not be used under the following condtions: 1. If a memory operand fetch is in progress 2. If an instruction fetch is in progress 3. If an RSTX transfer is in progress 4. If an ICT transfer is in progress The DI destination term sets the CPU DI full condition and clears any pending DI condtions that may be present.
7	MARIX	The Memory Address Register Indexed (19-bit condition causes the contents of the MIN bus (bits 08-31) to be presented to the Logical MAR. If the Nonextended mode is used, only bits 13-31 of the MIN bus are strobed into Logical MAR, bits 05-23. If the Extended Indexing mode is used, bits 08-31 of the MIN bus are strobed into Logical MAR, bits 00-23. If the Mapped mode is used during CREG+I, Logical MAR bits 04-08 address the MAP registers. At the end of CREG+I the contents of the MAP register are strobed into Physical MAR bits 00-08 to generate a 24-bit real address in Physical MAR (00-08) and Logical MAR (09-23). The content of the MIN bus is determined by the CPU Indexing algorithms. The ALU expressions for the Indexing algorithms are as follows: 1. MARIX=R(X)+10; 2. MARIX+R(DIX)+D1; Of the two expressions, the first expression is used for preindexing and the File register, addressed by 10 bits 09-10, is added to 10 bits 13-31. The ALU output is transferred to the MIN bus. If the File address provided by 10 bits 09-10 is equal to zero, the ALU Add function is changed to a Transfer B inputs to outputs function; thus 10 bits 13-31 are transferred to the MIN bus.

Table 3-8. Destination (D) Field (CROM20-23) Cont'd.:

Value	Syntax	Description
7 Cont'd.:		The second expression is used with postindexing and a memory indirect cycle must have previously occurred to load an indirect word in the DI register. In this case, the File register addressed by DI bits 09-10 is added to DI register bits 13-31. The ALU output is gated to the MIN bus. If the File address provided by DI bits 09-10 is equal to zero, the ALU add function is changed to a Transfer B inputs function; thus, DI bits 13-31 are transferred to the MIN bus. Either of the Indexing expressions cause the F- and C-bit calculations to occur according to the rules for macro-instruction memory addressing. The source for the Indirect bit and F- and C-bits is as follows: 1. B-mux bit 11 is the Indirect bit. 2. B-mux bit 12 is the F-bit. 3. ALU bit 30 is the C0 bit. 4. ALU bit 31 is the C1 bit. F- and C-bit manipulations are managed by the Virtual MAR. These bits are not transferred to Physical MAR until a SelBUS transaction occurs. The actual F- and C-bit configuration used may be modified by the 'FRCWORD' X-order and the 'FORCZF' Conditional order. If the MAR is used as a destination, and the output portion of a previously coded SelBUS transfer is not complete, the CPU enters an automatic WAIT state to prevent the destruction of the MAR contents, which must be valid for the successful completion of the SelBUS transfer.
8	R(Y)	The File Addressed by Register Number Field condition (File Write destination term) causes the output of the ALU to be strobed into the 32-bit T-register at the end of the CREG cycle. During the CREG+1 cycle, the contents of the T-register are strobed into the File register addressed by the Register Number field, CROM36-39. The 'Y' term may be any number or name, which is equated to a number, in the range of '0' to 'F'. Note: A File Write micro-instruction should not be directly followed by a File Read micro-instruction, since the File Write/Read actually occurs in the the CREG+1 cycle and a File/Write/Read would exist.

Table 3-8. Destination (D) Field (CROM20-23) Cont'd.:

Value	Syntax	Description
8 Cont'd.	R(Y)	If the 'OTHERBANK' X-order is used with a File Write function, it must be coded in the micro-instruction following the File Write micro-instruction. This coding is necessary so that the 'OTHERBANK' X-order is active during the CREG+1 cycle of the File Write instruction. Since the File Write function changes the contents of the T-register, if a File Write micro-instruction is coded before a previously coded Memory Write SelBUS transaction has been successfully completed, the CPU enters an automatic WAIT state until the Memory Write is complete. This WAIT state prevents the destruction of the T-register during the Memory Write transaction.
9	R(R)	The File Addressed by 10 bits 06-08 (File Write destination term) causes the output of the ALU to be strobed into the 32-bit T-register at the end of the CREG cycle. During the CREG+1 cycle, the contents of the T-register are strobed into the File register addressed by 10 bits 06-08. This destination term is used to coordinate the macro-instruction being executed and the File registers. Note: A File Write micro-instruction should not be directly followed by a File Read micro-instruction since the File Write actually occurs in the CREG+1 cycle and a File Write/Read conflict would exist. If the 'OTHERBANK' X-order is used with a File Write function, it must be coded in the micro-instruction following the File Write micro-instruction so that 'OTHERBANK' is active in the CREG+1 cycle of the File Write cycle. The File Write function changes the contents of the T-register. Therefore, if a File Write micro-instruction is coded before a previously coded Memory Write SelBUS transaction has been successfully completed, the CPU will enter an automatic WAIT state until the Memory Write is complete. This WAIT state prevents the destruction of the T-register during the Memory Write transaction.

Table 3-8. Destination (D) Field (CROM20-23) Cont'd.:

Value	Syntax	Description
A	None	Not Used
B	R(DIX) R(X) R(NCTR) R(S) R(RO) R(RD) FR(Z)	The File Register Address Selected by the File Read Select field (File Write destination term) condition strobes the output of the ALU into the T-register at the end of the CREG cycle. During the CREG+1 cycle the output of the T-register is strobed into the File register whose address is selected by the File Read Select field, CROM24-27. Refer to the File Read Select field (Table 5-9) description for a complete description of the Field addressing methods provided by these destination terms. For the 'FR(Z)' destination term, Extended Control, Register Select File addressing methods are used and the 'Z' term must be a File address in the range of '8' to 'F'. Note: A File Write micro-instruction should not be directly followed by a File Read micro-instruction, since the File Write actually occurs in the CREG+1 cycle and a File Write/Read conflict would exist. If the 'OTHERBANK' X-order is used with a File Write function, it must be coded in the micro-instruction following the File Write micro-instruction. This coding is necessary so that 'OTHERBANK' can be active during the CREG+1 cycle of the File Write micro-instruction. Since the File Write function changes the contents of the 7-Register, the File Write micro-instruction is coded before a previously coded Memory Write SelBUS transaction has been successfully completed and the CPU enters an automatic WAIT state until the Memory Write is complete. This WAIT state prevents the destruction of the T-register contents during the Memory Write transaction.
C	NU	The N-Counter Upper term causes the 8-bit N-Counter to be loaded with ALU bits 00-07 at the end of the CREG cycle.
D	NL	The N-Counter Lower term causes the 8-bit N-Counter to be loaded with ALU output 08-15 at the end of the CREG cycle.

Table 3-8. Destination (D) Field (CROM20-23) Cont'd.:

Value	Syntax	Description
E	FULLMAR	The Memory Address Register (24-bit) term causes the ALU output bits 08-31 to be strobed into the 24-bit Logical MAR at the end of the CREG cycle. During the CREG+1 cycle, the contents of Logical MAR (bits 00-08) are strobed into Physical MAR (bits 00-08), bypassing the CPU MAP algorithm. The FULLMAR destination term provides for the loading of a full 24-bit address into Physical MAR. Since two clocks are required to load Physical MAR, if the 24-bit address is required for a SelBUS transaction, the FULMAR destination term must be used in the micro-instruction prior to the SelBUS transaction term. If the MAR is used as a destination and the output portion of a previously coded SelBUS transfer is not complete, the CPU enters an automatic WAIT state to prevent the destruction of the MAR contents, which must be valid for the successful completion of the SelBUS transfer.
F	T	The T-register destination term causes the output of the ALU to be strobed into the 32-bit T-register at the end of the CREG cycle. Note: If the T-register is used as a destination and a previously coded SelBUS Memory Write is not complete the CPU will enter an automatic WAIT state to prevent the destruction of the T-register contents. The WAIT state is terminated when the Memory Write transaction is complete.

3.11 File Read Select (R) Field (CREG 24-26)

The R field (CREG24-26) indicates where the file address should be taken from in order to be able to read the 32 x 32 file. If we consider the file as 32 registers, then the R field of the microinstruction points to where we can get the register number from for reading the Register.

For example we want to Read either Register number 0 or 1 and bits 9 and 10 of DI register contain:

Bit 910
00
01

If the R-field contains the value 1 then any file read will read R0 or R1 depending on bits 9 and 10 of the DI register.

The following description gives the meaning of each bit combination and is summarized in Table 3-9.

Note that because of hardware restriction a file read should not follow a file write in microcode.

Table 3-9. File Read Select (R) Field (CREG24-26)

General: This field dictates the selection of the source of addressing the file for a read during this cycle. Since a write to the file is actually accomplished by writing 'T' into the file, during the cycle immediately following the DS cycle of the instruction requesting the write, care must be taken so that no attempt is made to read from the file in the next instruction following a write.

Influenced by: Control Orders REGSEL and MPROM.

Influences:

Value	Syntax	Description
0	R(REG NO)	Register Number is in CREG36-39.
1	R(DIX)D19-10	Register Number is in DI09-10. Register DI bits 9-10
2	R(X)109-19	Register Number is in 109-110 register I0 bits 9-10
3	R(NCTR)	Register Number is the ones complement of register N bits 4-7
4	R(R) 106-8	Register Number is in register I0 bits 6-8
5	R(S) 109-11	Register Number is in register I0 bits 9-11
6	R(RQ)106-7	Register Number is concatenation of regist I0 bits 6-7 and 'I'
7	R(RD)106-7	Register Number is concatenation of 1006-07 and DW.
		Note: This address may only be used with a LDMARIX order in the instruction proceeding it.
	EXTENDED REGSEL	Dependent upon Control Orders REGSEL and MPROM. T=6 or E. M=0.
	FR(REGNO) REGS8-F ONLY	The least significant 3 bits of the FR(2) field are initially Zero, with the most significant bit always a One.
	0	8
	1	9
	2	A
	3	B
	4	C
	5	D
	6	E
	7	F

3.12 Y-order (Y) Field (CREG 27-31)

The microinstruction bits 27-31 are decoded from the CREG to perform various control functions. These various functions and the bit combinations are shown in Table 3-10.

3.13 X-order (X) Field (CREG 32-35)

Like the Y-order field, microinstruction bits 32-35 are decoded from the CREG to perform various control functions in the data structure of the 32/75 computer. However this field is forced to NOP (inactive) if the T field has the value 3, 4 or 5. The various functions with the respective bit combinations are shown in Table 3-18.

Besides being used as the X-order field, mocroinstruction bits 32-35 are decoded for these other purposes:

a) U-Order
b) W-test
c) S-test.

These uses of bits 32-35 are explained in the following sections.

Table 3-10. Y-Order (Y) Field (CREG27-31) (2)

Influenced by:	None
Influences:	None

Value	Syntax	Description
0	NOP	This field is inactive, the default value
1	SHIFTS	This value causes the S-register to shift one bit position in the direction specified by the Shift Field. The vacated bit is filled as specified in the shift code.
2.	LITERAL (LONG) (LITL)	This value indicates that a long literal is to be used. The literal is constructed as described under the A-Mux section.
3	FILLEXP	This order is used in conjunction with floating-point and causes the output of the B-Mux to force BMUXO into positions BMUX01-07. This action eliminates the floating-point exponent and results in the ALU input being a correctly signed 32-bit mantissa.
4	BMGMR	This order gives control to the Bit Mask Generator (BMG) so that the byte decoded from I014-15 and/or the indexing logic can be selected. Bit selection within the byte is by decode of I006-08.
5	SETCAR	This order saves the carry-out of the ALU for later use in computing the most significant half of a doubleword operation.
6	USECAR	This order uses the previously saved carry (SETCAR) to supply the carry into the ALU.
7	WRPMAP	This order causes bits 16-31 of the file output to be written into the Protect register addressed by MAR05-08.
8	RELSW	This Read Least Significant Word order is used with the hardware Floating-Point.
9	RSTFP	This order is used to reset the hardware Floating-Point.
A	TSFILL	This order causes the sign bit (TREGOO) to be used as the fill bits when a T-register right nibble shift is selected by X-order NIBLT.
B	INCRN	This order causes the N-Counter (NCTR) to be incremented by 1.
C	RSTRHF	This order resets the Right Half Flag (RHFLG) flip-flop.
D	INHLHW	This order inhibits the write into bits 00-15 of the file.
E	INHRHW	This order inhibits the write into bits 16-31 of the file.
F	CLKDIV	This order causes SREGOO to be clocked into the least significant bit of the Quotient register.

Table 3-10. Y-Order (Y) Field (CREG27-31) (Cont'd)

Value	Syntax	Description
10	SELSPARE	This order selects the Quotient register on the Data bus in positions 16-31. Bits 00-15 are unspecified, and the ALU and scratchpad are disabled during this cycle.
11	RESETFF	This order resets all three groups of Level orders.
12	SAVESIGN	This order causes the sign bit of the ALU (bit 00) to be clocked into a flip-flop for later examination by the test condition structure.
13		Not Used
14	UPACK	User Panel Acknowledge (Clear Serial and Parallel panel interface).
15	CLRTO	This order clears the Bus Timeout flip-flops and sets the Ready Pending flip-flop.
16	RETAEXP	This order clears the Arithmetic Exception Pending flip-flop.
17		Not Used.
18-1F		These orders used in conjunction with CREG40-43 to control SelBUS transactions as follows:

Y-Order	CREG40-43	Transaction
19	8	FETCHPC (also increments PC by 4 or 1 word)
1C	1	ICT
1C	4	RSTX
1C	8	READ
1C	A	READ AND LOCK
1D		FETCHV (Fetch Instruction If Not Indirect, DRT →I1. Fetch Indirect Word If Indirect, DRT→DI).
1E	0	WDOT
1E	1	AICT
1E	4	ARSTX
1E	8	WRITE

Table 3-11. X-Order (X) Field (CREG32-35) (2)

Influenced by:	The Basic Test Codes 02, 4, 6-F, and U-Order flip-flop.
Influences:	
General:	The X-order does not exist if basic test codes 3 or 5 have been selected in this instruction. These bits address the W-test field (when selected) and the X-field is forced to a NOP. The Enable U-Order flip-flop must be reset (F/F=0).

Value	Syntax	Description
0	NOP	This field is inactive (default)
1	FRCWORD	The SEL transaction requested in this microinstruction is forced into the Word mode, regardless of the F- and C-bits of the computed address.
2	$TNIB_R^L$	This transaction causes the T-register (TREG) to be nibble shifted in the direction specified by the shift code. The vacant nibble is filled with zeros if the direction is left, and with the contents of SREG28-31 if the direction is right.
3	SHIFTDI	This transaction causes the DI bit to be shifted as determined by the shift code.
4	BLKCAR8	With this transaction, no carry propagation is allowed between ALU08 and ALU07.
5	TOGRHF	This transaction causes the RHFLG flip-flop to assume the state opposite its current setting.
6	SDEST	This transaction causes the output of the ALU to be transferred into the S-register (SREG). This action duplicates the choosing of the SREG as the Destination register in the Destination field. It also allows the SREG and another Destination register to obtain the ALU data simultaneously.
7	I1TOIO	This transaction transfers the contents of I1 into IO.
8	PCTOMAR	This transaction transfers the contents of the Program Counter (PC) into the virtual Memory Address Register (MAR).
9	SHIFTIO	With this transaction, IO 16-31 replaces IO oo-15 for execution of right halfword instructions. IO 16-31 receives IO 00-15 except for I-30 which receives a logical Zero.
A	RDLOCSTR	The Read Local Store order enables the scratchpad contents (required for a read or write) to be outputted onto the Data bus (DBUS). This ALU and SELSPARE commands are disabled during this cycle.
B	FIXEXP	This transaction is used with floating-point. If the T-register (TREG) is negative, the most significant byte of the ALU B input is complemented, and the three least significant bytes are passed straight through. The microword must supply the F=B function, and the result should adjust the exponent from positive to the sign of the result.

Table 3-11. X-Order (X) Field (CREG32-35) (Cont'd)

Value	Syntax	Description
C	SETRHF	This transaction causes the Right-Hand flag to be set.
D	OTHERBANK	This transaction causes a single cycle switch of the register bank setting. This switch is effective during the CREG cycle of the OTHERBANK command and therefore, effects the read specified in this command or the write specified in the previous instruction.
E	BMUXSE	This transaction causes bit 16 of the B-Mux output to fill bits 00-15 for sign extension of halfword operands.
F	BMUXZE	This transaction causes bits 00-15 of the B-Mux output to be unconditionally forced to Zeros.

3.14 U-order (U) Field (CREG 32-35)

When the Enable U-order flip=flop is set then microinstruction bits 32-35, decoded from the CREG are interpreted as the U-order. The meaning of the various bit combinations is then as given by Table 3-12 not table 3-11. This multiple use of the same field does not need to confuse the person writing microcode. One only need to be sure that the condition for interpreting the field in the way desired is fulfilled; the hardware takes care of the rest.

The enable U-order F/F is reset using microcode when H=5, P=0 and T=0.

See Figure 3-6 Below and microinstruction bits 36-39.

		P			C				H			
36	37	38	39	40	41	42	43	44	45	46	47	
1/0	0	0	0	-	-	-	-	0	0	0	0	

Figure 3-6 . Enable U-Order

*If bit 36=0 then the F/F is set, else it is reset!

Table 3-12. U-Order (U) Field (CREG32-35) (2)

General:		The Enable U-Order flip-flop must be set (F/F=1), otherwise the X-order test will be performed.
Value	**Syntax**	**Description**
0	NOP	This field is inactive.
1	RUN	This order causes the Run/Halt flip-flop to be set to the Run state.
2	HALT	This order causes the Run/Halt flip-flop to be set to the Halt state.
3	RESETIO	The Reset I/O order causes the SelBUS interface to poll the I/O reset line on the bus to reset the I/O controllers.
4	RSTPROTV	The Reset Protect Violation order resets the Protect Violation and MAP Invalid flip-flops.
5	LDMAP	The Load MAP order loads the MAP registers.
6	RDMAP	The Read MAP order enables the MAP mux (B-Mux).
7	LDWCS	The Load Writable Control Storage order loads the WCS registers.
8	RDWCS	The Read Writable Control Storage order enables the WCS Address Interdace flip-flops.
9	LDSTOP	The Load Stop order is used to clock the Stop Address compare "A" and "B" circuits.
A	RDLOCSTR	The Read Local Store order enables the output of the scratchpad (required for Read or Write) onto the Data bus (DBUS). The ALU and SELSPARE commands are disabled during this cycle.
B		Not Used
C		Not Used
D	OTHERBANK	This order causes a single cycle switch of the register bank setting. This switch is effective during the CREG cycle of the OTHERBANK command and, therefore, effects the read specified in this command or the write specified in the previous instruction.
E	BMUSE	The B-Multiplexer Sign Extend order causes bit 16 of the B-Mux output to fill bits 00-15 for sign extension of halfword operands.
F	BMUXZE	The B-Multiplexer Zero Extend order causes bits 00-15 of the B-Mux output to be unconditionally forced to Zeros.

3.15 W-Test (X) Field (CROM 32-35)

When the T field (CROM 00-03) has the value 3 or 5, then microinstruction bits 32-35 are interpreted as W-test field. This field is decoded from CROM. Both the X-order (Table 3-11) and the U-order (Table 3-12) are disabled when the W-Test is selected.

The various tests selected by the bit combinations is given in Table 3-13.

The Enable U-order flip-flop must be reset (F/F=0) and T=3 or 5 for the W-test to be executed.

Table 3-13. W-Test (X) Field (CROM32-35) (2)

Influenced by:	The Basic Test (T) Field (CROM00-03) values 3 and 5.
Influences:	
General:	The W-Test field is interpreted out of CROM and disables the X- and U-order fields during the DS cycle of the instruction.

Value	Syntax	Description
0	EXTLW	This condition is met if one of the following conditions exist: INSTTIMEOUT, OPNDTIMEOUT, INSTNORESP, OPNDNORESP, MUXINSTER, MUXOPNDER, NOTRUN, FFINT, FFSYSR, PWRFAIL, IPLSW, UPNLATTN, or PROTV.
1	SIGNSAVE	This condition is met if the sign saved during the last SAVESIGN order was negative.
2	ALUNEGW	This test is met if the sign bit of the ALU in the second preceding instruction was One (negative).
3	FFRUN	This test is met if the Run/Halt flip-flop is in the Run state.
4	BMUX00	This test is met if bit 00 of the B-Mux was Zero. Note: Although most tests cannot be performed in the instruction immediately following the creation of the value, because of the pipelining of the processor, the B-Mux tests indicated by * must be performed during the next executable instruction. Data being tested on the B-Mux is in complement form.
5	NORC	This test is met based on the contents of the PROM which examines ALU00-07.
6		Not Used
7	LFCIV	This test is met if bit 30 of the F- and C-bit extension of the MAR is a One. This bit normally corresponde to the most significant C-bit in a memory address.
8	BMUX16	This test is met if bit 16 of the B-Mux is Zero.
9	BMUX17	This test is met if bit 17 of the B-Mux is Zero.
A	BMUX18	This test is met if bit 18 of the B-Mux is Zero.
B	BMUX19	This test is met if bit 19 of the B-Mux is Zero.
C	LATERRW	This command indicates that a SelBUS operand fetch parity error or an arithmetic exception occurred during a previous instruction; or that an instruction fetch parity error or instruction has occurred with the current instruction.
D	DWORD	This test is met if the previous transaction was a double-word operand.
E	HWORD	This test is met if the previous transaction was a half-word operand.
F	BYTE	This test is met if the previous transaction was a byte operand.

3.16 S-test (X) Field (CROM 32-35)

CROM bits 32-35 are decoded as the S-test field when CROM 00-CROM 03 (T field) has the value 4. Only the true condition is tested in this field. The bit combinations that define the various tests are given in Table 3-17.

3.17 Summary of Use of Microinstruction Bits 32-35

We summarize the various conditions that are used to interpret microinstruction bits 32-35.

Conditions		Bit 32-35 Meaning
T field (CROM00-03)	Enable U-order F/F	
0, 1, 2, 6, 7	0	X-order
don't care	1	U-order field
3 or 5	0	W-test field
4	0	S-test field

Table 3-14. S-Test (X) Field (CROM32-35)

Influenced by:	The Basic Test Field (CROMOO-03) value of 4 only. There are no provisions for S-Test False.
Influences:	None

Value	Syntax	Description
0	%BLKTIMEOUT	No interrupt block mode timeout has occurred.
1	%WCS	Writable Control Storage not enabled.
3		Not Used
4		Not Used
5		Not Used
6		Not Used
7		Not Used
8	UNBLOCK	The external interrupts are not blocked.
9		Not Used
A	ENBL.AXEP	Enable Arithmetic Exception flip-flop is set.
B		Not Used
C	MODE75S	The CPU PSD mode flip-flop is set.
D		Not Used.
E		Not Used.
F	EXTMAPERR	The External MAP Error (Map Write Protect Violation) test is presently not being used.

3.18 Microinstruction Bits 36-39

Bits 36-39 of the microinstruction has multiple use depending on the value in the T, S, Y fields and CREG 44-47. These are summarized below:

Meaning of Bits 36-39	Condition
1. Z-test field	T=2, 6, A or E
2. Flip Flop Group 1 Field	C-order (CREG 44-47) =4
3. Flip Flop Group 2 Field	C-order =5
4. Flip Flop Group 3 Field	C-order =6
5. Extended Test (PC) Field (CROM 36-43)	T=1 or 9
6. FPU orders Group 2	T=D
7. Branch address (CROM 35-47)	S (CROM 04-06) $\geq$ 4
8. Literal generation (CROM 36-43)	Y (CREG 27-31) =02
9. CC Select (P) Field	M=1 or S $<$ 4, (CREG 44-47) =1 and the presence of a conditional test.

3.18-1 Z-Test (P) Field (CROM 36-39)

Microinstruction bits 36-39 are decoded from CROM as Z-test if the T field (CROM 00-03) has value 2 or 6. Table 3-15 gives the meaning of the various bit combinations.

Table 3-15. Z-Test (P) Field (CROM36-39) (2)

Influenced by:	Basic Test (T) Field (CROMOO-03) values of 2, 6, A, and E.
Influences:	
General:	This test is selected when primary test CROMO-3 specifies a Z-Test true or false.

Value	Syntax	Description
0	TRUE	This test is always met.
1	EXTL	This condition is met if one of the following conditions exists: INSTTIMEOUT, OPNDTIMEOUT, INSTNORESP, OPNDNORESP, MUXINSTER, MUXOPNDR, NOTRUN, FFINT, FFSYSR, PWRFAIL, IPLSW, UPNLATTN, and PROTV.
2	NHMPREV	This condition is met if the bit supplied during the last Multiply PROM order was Zero.
3	MODE75	This condition is met when the 75 Mode flip-flop equals One.
4	SIGNSAVEZ	This condition is met if the sign saved during the last SAVESIGN order was negative.
5	CCTEST	This condition is met if the user addressed test of the Condition Codes (CC's) is true.
6	LATERRZ	This condition, when met, indicates that a SelBUS operand fetch parity error or arithmetic exception occurred during a previous instruction, or that an instruction fetch parity error or instruction nonresponse (nonpresent memory) has occurred with the current instruction.
7	ALU4-7Z	This test is met if ALU bits 4-7 are Zero.
8	ALUNEG	This test is true if ALU sign bit (ALUOO) is One.
9	NCTRZ	This test is true if NCTROO-07 is Zero.
A	RHFLAG	This test is true if the Right-Hand flip-flop equals one.
B	NCTR4	This test is true if N-Counter bit 04 is One.
C	INDIR	This test is true if the indirect bit was set by the last MARIX Destination order.
D	NCTRO	This test is true if NCTR bit OO is One.
E	FCIV	This test is met if bit 30 of the F- and C-Bit Extension of the MAR is a One. This bit normally corresponds to the most significant C-bit in a memory address.
F		Not Used.

3.18-2 Extended Test (PC) Field (CROM 36-43)

When the T field has the value 1 or 9 bits 36-39 are combined with bits 36-43 to become the Extended Test (PC) field. This field is decoded from the CROM. The various tests selected by the bit combination are given in Table 3-16.

3.18-3 Flip-flop Field

When the C-order (CREG 44-47) has the values 4, 5 or 6, bits 36-39 is defined as the flip-flop field.

The flip/flops are divided into three groups:

CREG 44-47 = 4 selects groups 1
= 5 selects groups 2
= 6 selects groups 3

Bit 36 is used to set or reset the selected flip-flop while bits 37-39 are used as the address of the F/F. That is when

Bit 36 = 1 The F/F selected by bits 37-39 is set to 1
= 0 The F/F selected by bits 37-39 is reset to 0.

The various flip flops selected in each group are shown in Tables 3-17, 3-18 and 3-19.

3-18-4 FPU Orders Group 2 (CREG 36-39)

When the T-field (CROM 00-03) contains D (1101) microinstruction bits 36-39 are used for floating point operations. The various operations are defined in Table 3-20.

Table 3-16. Extended Test (PC) Field (CROM36-43) (2)

Influenced by:	The Basic Test (T) Field (CROMOO-03) values of 1 and 9.
Influences:	
General:	The Extended Test field allows "hops" or 12-bit externally supplied jumps based on a total of 40 separate tests. These 40 tests are further divided into eight groups of four. Within each group of five tests, individual lines or any OR subset of the give test lines may be tested.

BIT 36=0 enables the following tests addressed by bits 41-43:

Value	Syntax	Description
0	HIREG	The File Bank Control flip-flop test is true when the CPU is currently using the uppper file bank.
1	BADSCALE	This test is true if the Exponent field (bits 00-07) of the last ALU operation contained a significant mantissa bit.
2	IORESPRDY	The I/O Response Ready test is met for a single cycle as an IOM response to either an Advance Read Status Transfer (ARSTX) or an Advance Interrupt Control Transfer (AICT) sequence. This test indicates that the data requested has been assembled and that the CPU can enter an uninterruptible sequence to obtain the data from the IOM.
3	IONORESP	The I/O No Response flip-flop is set by the SelBUS interface to indicate to the system that the I/O transfer attempted had no response (i.e., there was no Transfer Acknowledge).
4	INSTNORESP	The Instruction No Response flip-flop is set by an instruction fetch to indicate to the system that the instruction attempted had no response.
5	OPNORESP	The Operand No Response flip-flop is set by an operand read to indicate to the system that the operand attempted had no response.
6	PROTV	The Protect Violate flip-flop is set when the hardware detects a write in protected memory and changes that write to a read. The flip-flop is reset by a Reset PROTV order. (V value of 4)
7	SONETO	S-register bit 0 is not equal to the T-register bit 0 (the sign bit is different).

BIT 37=0 enables the following tests addressed by bits 41-43:

0	PRIVBIT	Privileged Bit. When the Privileged Bit flip-flop is 0, the system is operating in the privileged state.
1	PWRFAIL	The Power Fail signal warns of impending power loss. This signal allows the system to place volatile information into core. The volatile information will be utilized when a restart command is executed.

Table 3-16. Extended Test (PC) Field (CROM36-43) (Cont'd)

Value	Syntax	Description
2	UPREQ	The User Panel Request signal indicates that the user panel (Turnkey Panel) has information to be transmitted to the CPU. The panel keeps this line high until it receives a User Panel Acknowledge (UPACK) signal. The UPACK signal indicates that the firmware has accepted the input.
3	IOCHBUSY	The I/O Channel Busy signal indicates that the response to an I/O transfer was "Channel Busy."
4	TO3SIG	The "To 3 Signal" test indicates that the values of TREG00 and TREG03 are different, implying significance to TREG03 (used for normalizing).
5	BIBUSY	The Bus Interface Busy signal comes from the SelBUS interface to the firmware. This signal indicates that the SelBUS interface has outstanding transactions to be completed/
6	LATERR	The Late Error signal indicates that a SelBUS operand fetch parity error or arithmetic exception occurred during a previous instruction; or that an instruction fetch parity error or instruction nonresponse (nonpresent memory) has occurred during a current instruction.
7	AEXP	The Arithmetic Exception condition is true when the Arithmetic Exception flip-flop is set.
	BIT 38=0 enables the following tests addressed by bits 41-43.	
0	EXFLAG	The Execute Flag flip-flop is used by the firmware to indicate that it is an execute command.
1	UARTTBMT	The UART Transmitter Buffer Empty condition indicates that the UART Transmitter buffer is empty and ready to receive additional information.
2	FFINT	The Interrupt flip-flop records the status of the external interrupt circuitry. This flip-flop is set by an Input/Output Microprogrammable Processor (IOM) winning the poll and requesting an interrupt. It is reset by either no one polling or the poll winner not requesting an interrupt.
3 4 5	IOTIMEOUT INSTTIMEOUT OPNDTIMEOUT	These three tests indicate that the SelBUS interface received a response in a read transfer, but an inordinate amount of time has passed without a Data Return Transfer (DRT).
6	FLAG	The Flag flip-flop is used by the firmware to flag internal events.
7	INTRENA	The Interrupt Enable flip-flop is set.
	BIT 39=0 enables the following tests addressed by bits 41-43.	
0	TRACE	The Trace flip-flop is a firmware flag used to trap the firmware out at location 0. This firmware flag has a variety of meanings, which are determined by the setting of additional flags maintained in the file.

Table 3-16. Extended Test (PCH) Field (CROM36-43) (Cont'd)

Value	Syntax	Description
1	PPATTN	The Serial or Parallel Control Panel Attention test indicates that the Serial or Parallel Control Panel needs service from the CPU.
2	IPLSW	The Initial Program Load Switch test indicates that the IPL switch has been depressed.
3	IORETRY	The I/O Retry test indicates that the response to an I/O transfer was a Retry.
4	INSTMIUERR	The Instruction Memory Interface Unit Error condition indicates that a parity error occurred during the execution of the current instruction.
5	OPNDMIUERR	The Operand Memory Interface Unit Error condition signifies that a parity error occurred during the execution of the preceding instruction.
6	EXTG	The External Global test indicates an External Global condition exists.
7	ADDRSTOP	The Address Stop condition indicates that the Serial or Parallel Control Panel has detected an address stop indication. This condition is caused by the Address Stop flip-flop being set.
	BIT 40=0 enables the following tests addressed by bits 41-43.	
0	UARTERR	The UART Error test indicates that a UART error has been detected.
1	UARTDAV	The UART Data Available test indicates that an entire character has been received and transferred to the UART Receiver Holding register.
2	FFSYSR	The System Reset flip-flop is set by a System Reset command. It remains set until the firmware issues a Clear Systems Reset (CLRSYSR) order. This condition indicates that the firmware has completed its portion of system initialization.
3	ENBL55	The Enable 55 option automatically enables the 55 mode. If this option is not used, the 55/75 mode is selected by IPL or Program Control.
4	SERIAL PANEL	The Serial Panel option allows information to be entered from the Serial Panel. If this option is not used, data must be entered from the Parallel Panel.

Table 3-16. Extended Test (PCH) Field (CROM36-43) (Cont'd)

Value	Syntax	Description
5	OPRNDPE	The Operand Parity Error test is used to perform operand error detection (operand parity error in-current instruction).
6	MAPINVALID	The MAP Invalid test is used when an invalid MAP condition is detected.
7	MAPMODE	The MAP Mode test indicates that the MAP mode is active.

Table 3-17. Flip-Flop Group 1 (C-Order=4) CREG 36-39 (2)

Flip-Flop Number	Micro Listing Functional Name	Definition
0	HIREG	When reset, selects register addresses 00-07, 08-0F. When set selects file register addresses 10-1F.
1	EXFLAG/EXFF	The execute Flag flip-flop is used by the firmware to indicate that it is an execute command.
2	PRIV	Privileged Bit: When reset, enables the Privileged mode. When set, enables the Nonprivileged mode (User's mode).
3	TRACE	Trace flip-flop: When set, indicates the contents of R (TRACE) are valid and causes an external global event.
4	DPEFF	Display Parity Error flip-flop (illuminates Parity Error indicator on the Serial Control Panel).
5	DINTRA	Display Interrupt Active flip-flop (illuminates Active Interrupt indicator on the Serial Control Panel).
6	DWAIT	Display Wait flip-flop (illuminates Wait Indicator on the Serial Control Panel).
7	ENAINTFF	Enable Interrupt flip-flop: When reset, internally inhibits interrupts. When set, internally enables interrupts.

Table 3-18. Alter Flip-Flop Group 2 (C-Order-5) (CREG36-39) (2)

Flip Flop Number	Micro Listing Functional Name	Definition
0	ENAUORD	Enable U-order when set.
1	UARTDS	UART data strobe.
2	UARTDAV	UART data available condition is set or reset.
3	MAPMODE	MAP mode operation is set or reset.
4	ENATBMT	The UART Transmitter Buffer Empty flip-flop is set or reset.
5		Not Used
6		Not Used
7	FLAG	Flag flip-flop (see Table 5-18 bit 38/6).

Table 3-19. Alter Flip-Flop Group 3 (C-Order=6) (CREG36-39) (2)

Flip-Flop Number	Micro Listing Functional Name	Definition
0	MODE75	75 mode flip-flop is set or reset
1	HUNBLOCK	Unblock flip-flop is set or reset.
2		Not Used
3	ENBL-AEXP	Enable Arithmetic Exception flip-flop is set or reset.
4		Not Used
5	ENASORD	Enable S-order flip-flop is set or reset.
6	FPDWORD	Floating-Point Doubleword flip-flop is set or reset.
7	DIS.BLKTIMEOUT	Disable Block Timeout flip-flop is set or reset.

Table 3-20. FPU Orders Group 2 (CREG36-39)

General:		This Group is enabled by a D in Test field bits 0-3.
Value	Syntax	Description
0	NOR1	This order is used to normalize an unnormalized answer.
1	NOR2	This order is used to normalize an unnormalized answer.
2	CORRZEROW	The Correct a Zero Word order is used when an operand equals 0. When this condition occurs, the number 40 is added as bias to the exponent field.
3	NOR4	No Operation
4	DSFPW	The Disassemble the Floating-Point Word order is used to load data into the Exponent and Fraction registers.
5	ASFPW	The Assemble the Floating-Point Word order is used to assemble the Floating-Point Word from the Exponent and Fraction registers for transmission to the CPU.
6	MASK	The Mask order is used to mask out bits not used for a specific operation. Bits 29-63 are masked for single-precision and bits 57-63 for double-precision.
7	PLUSONE	The Plus One order adds a carry bit to the exponent operation.
8		Not Used
9	FP.RST	This order is used to reset the Floating-Point.

3-18-5 CC Select Field (P) CREG 36-39

This field is decoded from CREG bits 36-39 when one of two bit combinations occur in a microinstruction.

1) When control-field CROM (07-09) is set to 1.

2) S-Field (CROM (04-06) does not contain any of the values 4, 5, 6, or 7; conditional-order (CREG 44-47) is set to 1; and the presence of a true test as defined by the T-field (CROM 04-03).

The various condition codes selected are defined in Table 3-21.

Table 3-21. CC Select (P) Field (CREG36-39) (2)

Influenced by:	Overlayable CREG. from Decode ROM
Influences	None
General:	This field exists when either of two situations occur. The first is the presence of Control Order 1 (overlay CC's); the second is the absence of Sequence Orders 4 through 7, the presence of Conditional Order 1 (Set CC's), and the presence of a true condition as defined by the basic test field and appropriate subtest fields (Z-Test W-Test, and Extended Test).

Value	Description	SYNTAX	CC1	CC2	CC3	CC4
0	Arithmetic & Logical	AL	Arith Ovflow	Result Pos	Result Neg	Result=0
1	Left Arithmetic Shift	V	Shift Ovflow	0	0	0
2	Masked Compare	E	0	0	0	Result=0
*3	1st Zero (for double operands)	D	0	0	0	Result=0
4	Bit Manipulation	BIT	Result≠0	Old CC1	Old CC2	Old CC3
5	S Reg	S	SRO1	SRO2	SRO3	SRO4
6	AEXP	AEXP	Old CC1	Result Pos	Result Neg	Result=0
7	Arithmetic Compare	C	0	A>B	A<B	A=B
8	Floating-Point	FP				
9-F	Not Defined					

*Note: Invoking Code 3 (1st Zero) is the for lower half of double word operations.

3-19. Microinstruction Bits 40-43

Bits 40-43 of the microinstruction has two meanings.

1) When the T field (CROM 00 - 03) equals 1 or 9, bits 40-43 are combined with bits 36-40 to define the extended test field.

Otherwise it is used as

2) Shift select (C) field and is decoded from CREG 40-43.

When used as shift select field, a complete description of this field is given in Table 3-22 (A) and (B).

3.19.1 Shift Select Field (C)

This field is decoded from CREG bits 40-43. A description of this field is presented in Table 3-22.

Table 3-22A. Shift Select (C) Field (CREG40-43)

Influenced by:	Overlayable CREG, from Decode ROM
Influences:	None
General:	This field exists when one or more of Y-Order 1 (shift S-bit), X-Order 2 (Shift T-nib), or X-Order 3 (shift DI bit) are present. It may not be present when either Sequence Order 5 (Br 8) or Sequence Order 6 (BR 12) is present unless Control Order 2 (OVLY shift) is also present. The shift code specifies the direction (R or L), mode (arithmetic or logical), and the fill information for S and DI. T is normally used only in the double mode, with S shifting by the A-mux; hence, the fill information for T is predetermined.

Bit Shift Table

Syntax	Code	Type	Reg Involved (*per X, Y ord*)	S-Fill	DI Fill
SLL	0	Left Logical singular	S or DI	'0'	*SR31
SLC	1	Left Circular singular	S or DI	SR00	*SR31
SLLD	2	Left Logical double	S and DI	DI00	'0'
SLCD	3	Left Circular double	S and DI	DI00	SR00
	4	*Left	S or DI	*SR31	*'1'
	5	*Left	S or DI	*DI31	*SR31
SLAD/	6	Left Arith double	S and DI	DI00	'0'
SLA	7	Left Arith singular	S or DI	'0'	*'1'
SRL/ SRLD	8	Right Logical singular or double	A and/or DI	'0'	SR31
SRA/ SRAD	9	Right Arith, singular or double	S and/or DI	SR00 (sign)	SR31
	A	*Right	S and DI	*DI00	'0'
	B	*Right	S and DI	*DI00	*SR00
SRC	C	Right Circular singular	S and DI	SR31	*'1'
SRCD	D	Right Circular double	S and DI	DI31	SR31
	E	*Right	S or DI	*DI00	*'0'
	F	*Right	S or DI	*'0'	*'1'

*Note: These codes and associated fill information are not used for 86 Emula

Table 3-22B. NIB Shift Table

Note that the S-Reg Nibble shift is accomplished by the A-mux paths indep endently of the Shift Code and the Shift T-order.

Shift Code	Type	T-Fill Nibble
0 through 7	Left	'0'
8 through F	Right	SR28-31
A-mux Code	**Type**	**S-Fill Nibble**
2	Left	TR00-03
3	Right arithmetic	SR00 (sign bits)

Additional Left Shift S-Reg Path (A-Mux Code 1)

Assembler Syntax	Shift Code	Type	A-Mux 31 (Fill Bit)	
SLL	0, 8	Left Logical singular	'0'	A-Mux 00-30=SR01-31
SLC	1, 9	Left Circular singular	SR00	" " " "
SLDD	2, A	Left Logical double	DI00	" " " "
SLCD	3, B	Left Circular double	DI00	" " " "
--	4, C	*Left	*DI31	" " " "
--	5, D	*Left	*DI31	" " " "
SLAD	6, E	Left double arithmetic	DI00	" " " "
SLA	7, F	Left double arithmetic	'0'	" " " "

*Note: These codes and associated fill information are not used for 86 Emulation.

3-20. Microinstruction Bits 44-47

Bits 44-47 of the microinstruction play a dual role:

They are used as

1) The Conditional Orders (PCH) field

or 2) The FPU orders Group 1.

3-20-1. Conditional Orders (PCH) Field (CREG 44-47)

If bit 04 of the S-field (CROM04-06) is set to 0 and T field does not equal to 8 or D then microinstruction bits 44-47, decoded from the CREG are interpreted as the Conditional Order field.

The various conditional orders are described in Table 3-23.

The relevant microinstruction bits are illustrated in Figure 3-7

Table 3-23. Conditional Orders (PCH) Field (CREG44-47)

Influenced by:	Basic Test (T) Field (CROMOO-03) and Sequence Control (5) Field (CROM04-06) Value of 0 through 3.
Influences:	
General:	The Conditional Order field exists if the Sequencing field contains codes 0 through 3, and the order is issued only if the test condition was met during the CROM cycle of this command. The S-Field must not equal 4 → 7.

Value	Syntax	Description
0	NOP	This instruction implies that no sequence change is to be made, regardless of the addressed test condition status. The Conditional Order field is the only part of the circuit which uses the test result.
1	SETCC	This instruction conditionally causes the Condition Codes to be set as requested by the CC Select Field (See Table 5-23).
2	FORCEFZ	This instruction conditionally forces the F-bit to Zero in the current memory read. It is used by floating-point commands where the F-bit is part of the op code.
3	CLRSYSR/ IGNSTOP	This instruction conditionally clears the System Reset flip-flop and sets the Disable Address Stop flip-flop.
4	ALTERFF1	This order conditionally causes the setting or resetting of one of the eight General Purpose (GP) flip-flops. The Flip-Flop field (CREG36-39) provides the input to the flip-flop in bit 36, and flip-flop address in bits 37-39. (Group 1)
5	ALTFRFF2	This order conditionally causes the setting or resetting of GP flip-flop 2. (Group 2)
6	ALTFRFF3	This order conditionally causes the setting or resetting of GP flip-flop 3. (Group 3)
7	CLRS	This order conditionally clears the S-register.
8	ABSDI	This order, in conjunction with a microstep of DEST=0+DI, changes the function to DEST=0-DI if DIOO is One; therefore, the ALU output is the absolute value of DI.
9	ABST	The Absolute Value of the T-register order, in conjunction with a microstep of DEST=0+TREG, changes the function to DEST=0-TREG if TREGOO is One; therefore, the ALU output is the absolute value of TREG.
A	DIVIDE	This order changes the ALU function from + to - if S-register bit 0 is Zero.
B	MPY	This order conditionally changes the ALU function from + to - if HMPREV is One.
C	SETXCC	This order conditionally sets the Condition Codes from S-register bits 1-4 and sets the Extended Address mode from S-register bit 5.
D	CLDNU	This order conditionally loads the N-Counter from ALUOO-07.
E	CDECRN	This order conditionally decrements the N-Counter.
F	SETAEXP	Set Arithmetic Exception flip-flop conditionally.

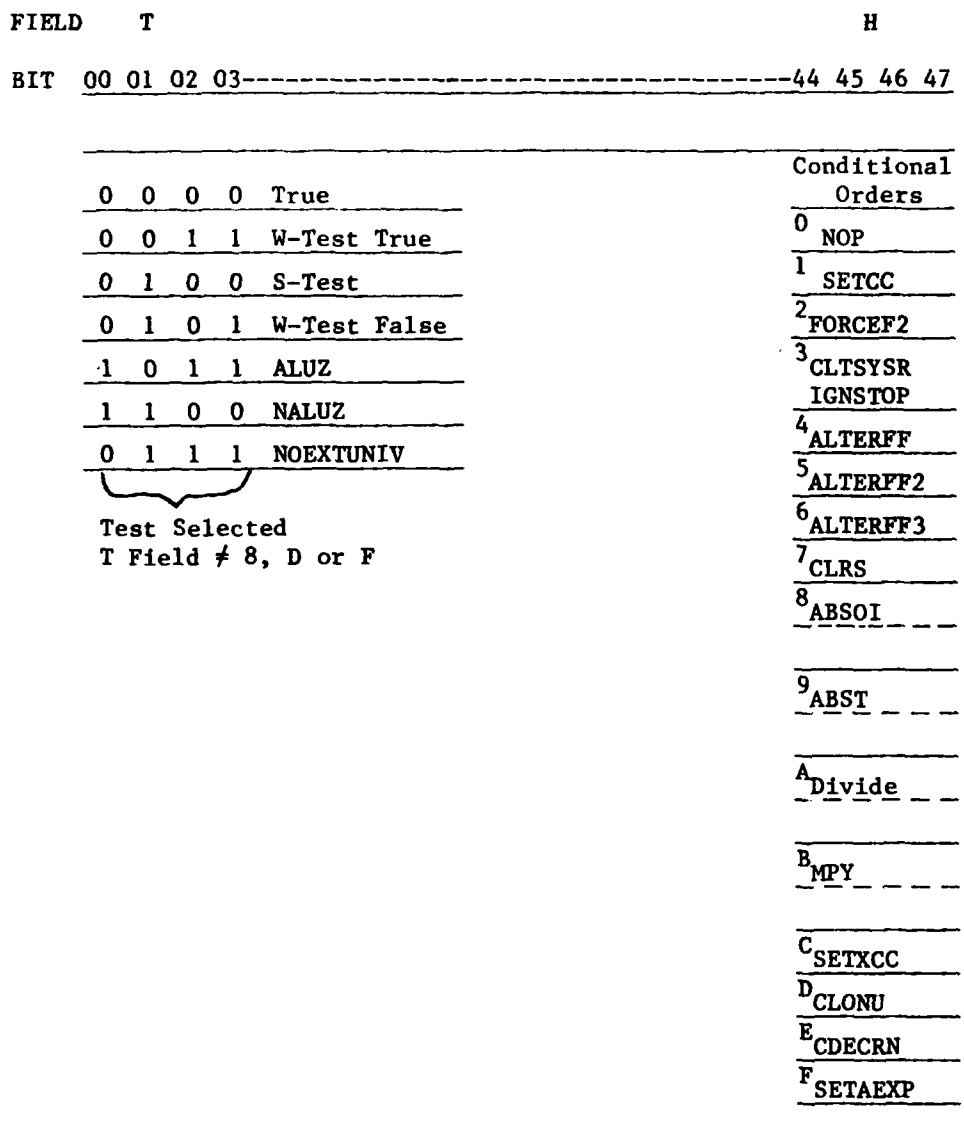


Figure 3.7 . Conditional Orders (General)

3-20-2. FPU ORDERS GROUP 1 (CREG44-47)

When the T-field (CROM 04-03) contains the value 8, D or F, bits 44-47 become the FPU Orders Group 1.

Table 3-24 gives the meaning of each bit combination in this mode.

Table 3-24. FPU Orders Group 1 (CREG44-47)

General:		This group is enabled by either 8, D, or F in Test Field bits 0-3.
Value	Syntax	Description
0	NOP	This field is inactive.
1	RND	The Round order causes a One to be added to the Normalized answer.
2	CORRMNG	The Correct Maximum Nagative order corrects the answer if a -1 has occurred as an answer.
3	DELSHF	This order speeds execution of add and subtract operations by using only hardware methods. This order operates in four steps: 1. It takes the difference of exponents. 2. It shifts the mantissa with the smaller exponent. 3. It adds the mantissas together. 4. It checks for an overflow condition and corrects that condition, if necessary.
4	OVFMOPEZ	This order is used by the divide instruction. It sets the overflow flag when the divisor is equal to Zero.
5	COMPLAN	This order is used when the operand is negative. When this condition occurs, the B-register is subtracted from Zero.
6	COMPLAP	This order is used when the operand is positive. When this condition occurs, the B-register is subtracted from Zero.
7	COMPLAF	This order is used to complement the answer when the sign flip-flop is set during a Multiply or Divide operation.

References

1. WCS Users Manual
Publication No. 301-322344-00, March '79
Systems Engineering Lab. Inc.

2. Technical Manual
32/70 Series Computer. Publication
No. 303-320070-000, April '79

3. SEL32 Technical Manual
CPU Text Vol. 1-3. Publication
No. 303-322000-000, April '76.
Systems Engineering Lab.

Appendix

Included as an Appendix to this report is very useful information on microcoding. These are taken from reference number one and are organized as follows;

Appendix A: Firmware coding

Appendix B: WCS Firmware Techniques

Appendix C: WCS Sample Programs

Appendix A

FIRMWARE CODING

INTRODUCTION

It is preferred that the beginning Microprogrammer initially write firmware in a serial manner, with one operation per microword statement. This allows for easier debugging of the procedure, and when the logical operation has been verified to be correct, then statements can be combined for parallelism to reduce execution time.

The beginning firmware programmer will find it enough of a task to learn the Data Structure operation and control. The primary obstacle to exercising the parallel power of the 32 SERIES of Computers supporting the WCS option is the problem of field conflicts within the microword due to multiple usage of some of the fields. The Micro-Assembler will detect and report any field conflicts for the user who attempts to utilize the parallelism. However, in disentangling these, the user risks generating less efficient programs than if he had proceeded in the recommended direction of serial coding first, and then looked for the obvious opportunities for parallel processing.

Control of the Data Structure often requires painstakingly small steps. For instance, defining a 32-bit constant value using a literal generator can take multiple micro-steps. This is due to the limitation of only being able to specify one 8-bit byte at a time to the Literal Generator. In particular, a 4-step sequence, similar to the following, is required to load the T-register with the hexadecimal value 12345678:

```
T=@12000000;
T=@00340000+T;
T=@00005600+T;
T=@00000078+T;
```

In some special cases, more than one byte may be defined in a single statement by making use of the ALU functions +1 and -1. For example,

```
T=@00100000-1;
```

will produce the 20-bit hexadecimal mask 000FFFFF.

```
T=@FF7FFFFF+1;
```

will produce the hexadecimal mask FF800000.

This level of detailed specification is only one aspect of concern to the Microprogrammer. Another significant concern is the element of timing. Referring back to the description of the CROM and CREG cycles of Micro-Instruction execution, the user can understand why the following sequence is valid for checking if the S-register is Zero.

```
1.  NOD=S;
2.  *NOP;
3.  IF NALUZ *GOTO SNONZERO;

Cycle 1:  CROM 1    Implied Primary Test "If True" is successful

Cycle 2:  CREG 1    Data From S-register gated to ALU

          CROM 2    NOP
```

Cycle 3: CREG 2 NOP

CROM 3 Test state of ALU output, select next CROM address

Cycle 4: CREG 3 NOP

CROM 4 or CROM of SNONZERO

Thus we see that ALU tests must be specified in the statement two instructions later than the statement which specified the ALU function being tested.

B-mux tests differ in two ways; the state is available for testing one cycle after the B-mux data is specified, and the B-mux is one of several inverting elements in the Data Structure, which requires that the test be inverted. To determine if the sign bit is set in a halfword value held in the T-register, the statement sequence might be:

```
1.  BMUX=T;
2.  IF %BMUX16 *GOTO SIGN.BIT.SET;
```

Cycle 1: CROM 1 Implied Primary Test "If True" successful

Cycle 2: CREG 1 Data from T-register gated to B-mux

CROM 2 Test B-mux output state, select next CROM address

Cycle 3: CREG 2 NOP

CROM 3 or CROM of SIGN.BIT.SET statement

This example also demonstrates that a B-mux selection can be specified independently of any ALU operation, and in fact, both may be specified in a single statement:

```
BMUX=T,DI=S+1;
```

Another situation where timing of the Microengine is significant, is in the use of the General File register. The following statement is possible.

```
R(1)=S+R(1);
```

That is, the same register may be specified as a source of data and as a destination in a single statement in spite of the fact that the General File registers cannot be Read and Written at the same time. This is possible because the source file is selected in the CROM cycle, the data is gated from the General File register, the ALU function is performed, and the ALU output gated to the T-register in the CREG cycle (refer to the Data Structure, Figure 5-47) and the data is gated into the File register from the T-register on the CREG+1 cycle.

Some of the implications of this General File register timing, which must be considered, are:

1. A General File Read statement may not follow a General File Write statement. This is because data cannot be read from and written into the General File registers simultaneously. For a Write statement, the data is stored in the General File Register in the CREG+1 cycle (which is the same as the CREG cycle of the following instruction) so the General File register may not be sourced in the following Micro-statement. The Micro-Assembler does not detect or flag an error of this type since statements are independent items to the assembler.

AD-A106 779 FLORIDA INST OF TECH MELBOURNE DEPT OF ELECTRICAL AN--ETC F/G 14/2
IMPLEMENTATION OF THE RECOMMENDATIONS MADE ON THE TECHNICAL REP--ETC(U)
JUN 81 J HADJILOGIOU AFOSR-81-0120

UNCLASSIFIED AFOSR-TR-81-0704 NL

CONT

The data written into the General File register on the CREG+1 cycle is available to be sourced from the T-register on any cycle after the CREG cycle, specifically including the CREG+1 cycle. That is,

```
R(1)=S+R(1);
R(2)=1+T;
```

produces the same results as

```
R(1)=S+R(1);
*NOP;
R(2)=1+R(1);
```

2. The file selection may be specified in the data structure rather than in the Micro-Instruction:

```
R(R)=S+R(R);
```

where the (R) indicates that the file number is specified by the data in the IO register, bits 06-08. The File register selection is accomplished in the CROM cycle, which implies that the proper value must be established in IO prior to this time. This requires that the order I1TOIO must have been issued sometime previous to, but not in, the preceding statement. An I1TOIO order in the immediately preceding statement would be effected in that statements CREG cycle which overlaps the CROM cycle of this statement, and would not be correct for selecting the General File register in this statement.

3. The 32 General File registers are divided into two banks of 16 registers each, addressed as 00-0F, with the HIREG flip-flop defining which bank of 16 registers.

```
R(1)=S+R(1);
```

would source from and store data into File register 1 of the Lower Bank if HIREG is not set, and register 1 of the Upper Bank if HIREG is set. It is important to note that the HIREG flip-flop must be set or reset more than one statement prior to this statement, as the setting of the flip-flop is done on the CREG cycle. Assuming that the HIREG flip-flop is reset, the following statement sequence:

```
SET(HIREG);
R(1)=STR(1);
```

will result in File register 1 on the Lower bank being sourced and added to the S-register with the result stored in File register 1 of the Upper Bank, because HIREG flip-flop was reset at the beginning of the CROM cycle, but set on the CREG cycle of the second statement. If the intention is to source the File register in the Upper Bank in this situation, it could be accomplished as follows (assuming that HIREG is reset):

```
SET(HIREG);
R(1)=S+R(1),OTHERBANK;
```

The OTHERBANK order is effective in the CROM cycle, when the File register 1 is selected, so the Upper Bank R1 is sourced. In the CREG cycle, when the File Write is selected, the OTHERBANK is not in effect, but HIREG is set, so the File Write goes into the Upper Bank also.

If HIREG is reset and the Upper Bank register 1 is to be modified, the following sequence allows this without altering the HIREG flip-flop:

```
R(1)=S+R(1),OTHERBANK;
OTHERBANK;
```

In this case, the second OTHERBANK Order is in effect during the CREG cycle of the first statement, when Destination register selection is accomplished. A summary of the timing for this statement:

R(1)=S+R(1);

CROM CYCLE;

File selection of register 1 for source utilizing the state of the HIREG flip-flop and the OTHERBANK signal if specified in this statement.

CREG cycle:

Data from selected File register added to S-register contents, ALU output stored in the T-register, file selection of Register 1 for Destination, utilizing HIREG state and OTHERBANK if specified in the following statement.

CREG+1 cycle:

Data is strobed from the T-Register into the File register selected during the previous cycle.

INTERLOCKS

The Register Interlocks are another timing consideration. These are hardware imposed interlocks on the MAR, T-, DI, and I1 registers, which inhibit access to these registers while certain bus transactions are in process. An instruction attempting to access one of these registers, when an interlock is in effect, will "hang" for one or more machine cycles, until the interlock condition is ended.

The interlock is imposed on loading the MAR register when any Memory Read or Write bus transaction is initiated; this interlock will inhibit any micro-order attempting to load the MAR (such as MAR, FULLMAR=, MARIX=, or FETCHPC) until the bus transaction is complete, whether successful or not. This period is normally two clocks, but may be longer due to heavy bus activity or bus memory.

An interlock on loading the T-register is imposed for any Write bus transaction, from the clock following the CREG cycle of the transfer request, to the clock following the Transfer Acknowledge on the bus. This interlock will block any attempt to change the T-register (such as T=, TNIB shift, or a General File Write).

An interlock is imposed on sourcing the DI-register for any Read bus transaction, from the CREG cycle of the Read request until one clock after the data is strobed into the DI register. This interlock inhibits any sourcing of the DI register through the B-mux only. SHIFTDI orders and use of DI contents for File register selection are not inhibited by the interlock.

The I1 interlock is similar to the DI interlock, but is imposed for any instruction fetch from memory. When in effect, it blocks the I1TOIO order and the JUMPZ (which implies an I1TOIO order).

RIGHT-HAND FLAG

The Right-Hand Flag flip-flop is a hardware assist to aid in processing the halfword instructions in the 32 SERIES Macro-Instruction set. The state of the Right-Hand Flag (RHFLAG) affects several events:

1. All bus transfer requests are ignored if issued when RHFLAG is set.
2. The I1TOI0 order is not effective when RHFLAG is set.
3. The SHIFTIO order is not effective when RHFLAG is reset.

In addition, TOGRHF does not execute if the right half of IO is a NOP instruction and the EXFLAG flip-flop is set.

DATA SHIFTING

The 32 SERIES of Computers supporting the WCS option provides for a variety of data shifting in four registers. The are S-, DI, T-, and IO registers. In addition to these, the B-mux supports halfword swapping of the General File registers, and the A-mux provides for an input of shifted data from the S-register. Some of the registers may be coupled for doubleword shifts.

The S-register can be shifted one bit Left, Right, or Circular by a SHIFTS Y-Order.

The DI-register can be shifted one bit by a SHIFTDI X-Order, but the DI-register is not normally used for single register shifting because the carry-in or fill bits are not what would normally be expected.

The S and DI-registers may be combined as a left and right pair for doubleword shifting with the syntax element SHIFTD. This 64-bit shift may be left, right, or circular, specified by a shift code enclosed in parenthesis following the shift order. For example,

```
SHIFTS(SRA);
```

specify shifting the S-register right one arithmetic bit.

```
SHIFTD(SLCD);
```

specify shifting the S and DI-registers as one 64-bit register left circular one bit.

The general form for the shift code is:

$$\left(S \quad \left\{ \begin{matrix} \text{Left} \\ \text{Right} \end{matrix} \right\} \quad \left\{ \begin{matrix} \text{Arithmetic} \\ \text{Circular} \\ \text{Logical} \end{matrix} \right\} \quad [\text{Double}] \right)$$

where only the "D" for double is optional. The fill bits for the various shift codes are:

SHIFT CODE	S-FILL	DI-FILL
SLL	"0"	*
SLC	S00	*
SLLD	DI00	"0"
SLCD	DI00	S00
SLAD	DI00	"0"
SLA	"0"	*
SRL	"0"	S31
SRLD	"0"	S31
SRA	S00	S31
SRAD	S00	S31
SRC	S31	*
SRCD	D131	S31

*= Unspecified

Appendix B

WCS FIRMWARE TECHNIQUES

INTRODUCTION

The Microprogrammer writing a WCS routine is able to achieve great program efficiency because his program is in direct control of the Data Structure of the computer CPU. This superior power is balanced by the critical responsibility of carefully exercising that control. Once control of the CPU has been passed to a user program in WCS by means of a JWCS Macro-Instruction, the WCS routine is in total control of the computer. The WCS program directs the entire Data Structure until control is yielded back to the CPU CROM firmware. When running a WCS program, there is nothing in the computer to protect it from any program errors in the WCS program, or take over if the program should run improperly. Any WCS programming error is likely to result in a machine "Hang" condition which may only be recovered from by a Power Down/Power Up sequence.

STATE OF DATA STRUCTURE

The WCS programmer needs to be aware of two general areas: the state of the Data Structure when the user obtains control, and the consequences of any changes to the Data Structure state made under WCS control. When a WCS Microprogram is entered from a JWCS Macro-Instruction, the following conditions are known:

1. HIREG flip-flop is reset.
2. ENAUORD flip-flop is reset.
3. EXFLAG flip-flop is reset.
4. RHFLAG flip-flop is reset.
5. Register IO contains the JWCS Macro-Instruction.
6. Register I1 contains the Macro-Instruction in the memory location following the JWCS.
7. Register PC contains the address of the memory location two words past the JWCS.
8. Register MAR contains the final computed effective address of the JWCS instruction.
9. Register T contains the least significant 6 bits of the MAR, left zero filled.
10. Register T contains the hexadecimal value 1000 added to the contents of the T-register.
11. Register DI, only if the JWCS instruction specified an indirect address, contains the effective address of the JWCS instruction prior to any post-indexing adjustment.

The following general purpose registers have dedicated usage, and should not be altered:

BANK	FILE	LABEL	CONTENTS OR USAGE
Lower	8	PCMASK	0007FFFC
Lower	9	STMASK	FE000000
Lower	12	ALEVEL	Highest active interrupt
Lower	15	TRACE	Status flags
Upper	15	ZERO	0

In addition, if MAPMODE is in effect, Lower Bank general file registers 10, 11, and 13 should not be altered.

The Microprogrammer coding WCS routines may alter the contents of the IO, MAR, DI, I , and S-registers without concern. Any changes made to PC or to I1 must be compensated for either by restoring the original values, or by substituting other valid values. At the time of exit from the WCS routine, I1 should contain the next Macro-Instruction to be decoded and executed, and PC should contain the Macro address of the next following instruction. The Microprogrammer should check and save the state of any of the flip-flops which his process is required to alter, and he should not fail to restore those flip-flops to their original state prior to his exiting the WCS routine.

WCS ENTRY

The entry to WCS is accomplished by a JUMPS Micro-Instruction in the CPU firmware segment which processes the JWCS Macro-Instruction. Thus, upon entry to WCS, the S-REG contains the WCS entry point, which is 1000_{16} greater than the WCS address specified in the JWCS Macro-Instruction. This reflects the fact that the WCS is viewed as an extension to the CROM by the Microprogrammer, with addresses ranging from 1000 to 1FFF added onto the CROM addresses 000 to FFF. This is in contrast to the fact that a macro-level programmer views WCS as a separate entity, with addresses ranging from 000 to FFF. If the macro-level programmer does specify an address in the range of 1000 to 103F in a JWCS instruction, the correct entry point is computed because only the least significant bits are utilized in computing the entry point.

The CPU firmware segment which processes the JWCS Macro-Instruction also contains a PUSHJ Control order, which stores in the first level of the J-Stack a CROM address to which the WCS programmer may return. This return is accomplished with a JUMPJ Sequence order. At the return CROM location, the CPU firmware initiates a load of the instruction pipeline register (I1), and sets up to decode the next Macro-Instruction, which is currently in the pipeline register (I1). Maintaining this CROM return address in the J-Stack effectively reduces from three to two the number of J-Stack levels available to the WCS programmer. Thus he may link to and return from subroutines only two levels deep. These subroutines may be in either WCS or CROM areas, as the J-Stack functions with 13-bit CROM addresses.

USEFUL CROM SUBROUTINES

Some useful subroutines exist in the CPU firmware, which may be utilized by the WCS Microprogrammer. In order to make use of any of these routines, the user must know the exact CROM address of the routine, and assemble this address into the user WCS routine with an $EQ statement. The addresses of these routines can be obtained from the CPU Firmware Manual. These routines are:

1. CD.DUD — Returns immediately by means of a JUMPJ; essentially, a one-cycle delay.
2. DOUBLE.DUD — A two-cycle delay.
3. TRIPLE.DUD — A three-cycle delay.
4. TEST.BIBUSY — Returns one cycle after last outstanding bus transaction completes.
5. DELAYED.BI.BUSY.TEST — Enters TEST.BIBUSY after a one-cycle delay.

6.	MEMORY.READ	Initiates a 32-bit memory read from the address contained in the MAR register; returns after data is returned to DI register.
7.	MEMORY.WRITE	Writes contents of T-register to memory address designated by contents of MAR register; waits for completion.
8.	READ.WCS	Reads an address in WCS from WCS.

Note

To use Write WCS, address is placed in the S-register and Data is in the T-register. In the case of Read WCS, the address is placed in the S-register and data is returned in the S-register.

9.	WRITE.WCS	Writes an address in WCS from WCS.

CAUTIONS

Certain orders should not be used in WCS firmware. These are the Extended Control orders: REPEAT, SCALE, and NORM. These particular orders require a relatively large portion of the machine cycle time to execute. When combined with the time required for the RAM access to fetch the Micro-Instruction from WCS, these orders may not complete within the 150-nanosecond time limitation. The Read/Write of WCS from WCS cannot be accomplished using the U-Orders.

MACRO-/MICRO-LEVEL COMMUNICATIONS

Several alternatives exist for communication of data between the Macro-level programs (software) and the Micro-level routines (firmware).

The most direct method of communicating data or data addresses is through use of the general purpose registers. These registers are accessible by the software as R(0) through R(7). They are also accessible by the software as R(0) through R(7). They are also accessible to firmware as file registers 0 through 7 of the lower bank. This dual accessibility provides a direct means of transferring data or data addresses between software and firmware. There is also the implication that the firmware programmer in WCS should be aware that by altering any of these eight registers in his routine, he could adversely affect the software program which called him.

Another means of communication is by the Condition Codes. The state of these four bits may be set in firmware by means of any of the SETCC orders. Upon return to the software level, the state of these four bits may be tested indivudually or in combination. Thus the condition codes may be utilized for transferring limited information from WCS routines to macro-level programs.

In the case where the amount of information to be transferred exceeds the capability of these methods, a useful technique would be the argument list. In this method, the macro-level programmer creates a sequential list of data and/or data addresses in memory. The location of this list may be predefined and thus know to the firmware routine, or it may be relative to the current contents of the PC register, or it may be passed to the firmware routine in one of the general purpose registers. The firmware routine in WCS may then acquire the data it requires by means of successive memory accesses. It may also store computed results in allotted list entries, or in memory locations designated by a list entry.

Another possible means of passing data from software to firmware is within the JWCS statement itself. Since the JWCS op code, index register designation, and indirect bit occupy bits 0 through 11 in the instruction, and the WCS entry point occupies only bits 26 through 31, the programmer has available bits 12 through 25. Information in these bits is available to the firmware routine because the entire JWCS instruction is held in the Decode register (10).

Appendix C

WCS SAMPLE PROGRAMS

INTRODUCTION

This section contains a collection of WCS coded routines for use in assisting the user in writing Microprograms in WCS.

SAMPLE PROGRAM 1

This program function is to invert the order of bits in a general purpose register; e.g., to put the value of bit 00 in place of bit 31, the value of bit 01 in place of bit 30, and so forth until all 32 bits have been moved. This function is used in some signal processing applications and in machine conversions.

The algorithm is straightforward. Each bit is checked, copied into the result, and then shifted in opposite directions. A flowchart representation of the program is presented in Figure 10-1. The first and final program generations are shown on Figures 10-2 and 10-3.

The N-Counter, the obvious choice for count control, is so used here. Less obvious is the choice of registers to shift.

The choice of registers is based on the fact that specific bits may be checked in the B-mux in the first Micro-Instruction following a data transfer through it. Referring to the Data Structure (Figure 5-2), only the DI and T-registers are available for use. It is desired to shift 1 bit at a time, so the selection is limited to the DI register. The T-register is directly shiftable only by 4-bits at a time. The target value register is then selected. The T-register and the S-register are available, but since the S-register is the only one capable of 1-bit shifts, it is the register selected. The T-register is then left to hold a single "ON" bit for generation of the target.

In the implementation of this program, the first three Micro-Instructions (refer to the bit-swap program following this discussion), lines 12, 13, and 14, are used to set up initial conditions in the structure. The use of NU= at line 13, rather than NL= is merely an historic convention.

Inside the loop, lines 15-18, the input data from the DI register is gated so it can be checked quickly. In line 15, the DI register is shifted, which takes place independently of and apparently after the NOD=DI transfers. Checking further in the flowchart (Figure 10-1), it is discovered that the noncompeting, invariant operation N-1 — N (Bubble J) could be executed in the same Micro-Instruction with Bubbles E and F. The shift of the DI register is Left Logical, which disregards the old bit 00 and shifts a Zero into Bit 31.

In line 16 the old DI register bit 00 is checked, having been Zero as it was transferred through the B-mux in the preceding Micro-Instruction. Note here that the condition is true and thus the branch is taken when BMUX00 is Zero (this is an opposite sense to the typical logical test).

Since the S-register is cleared in line 14, it can be ignored if it is desired to place a new Zero into the S-register. Otherwise, a single bit is copied from the T-register into the S-register, bit 00. This is done by using the ALU, A-mux, B-mux, and the Destination order S=S: T, which merely reloads the S-register with the logical summation of the S- and T-registers.

Under this plan, the new bit 00 data is moved further and further to the right until the S-register is filled. Line 18 accomplishes this plan by shifting the S-register Right

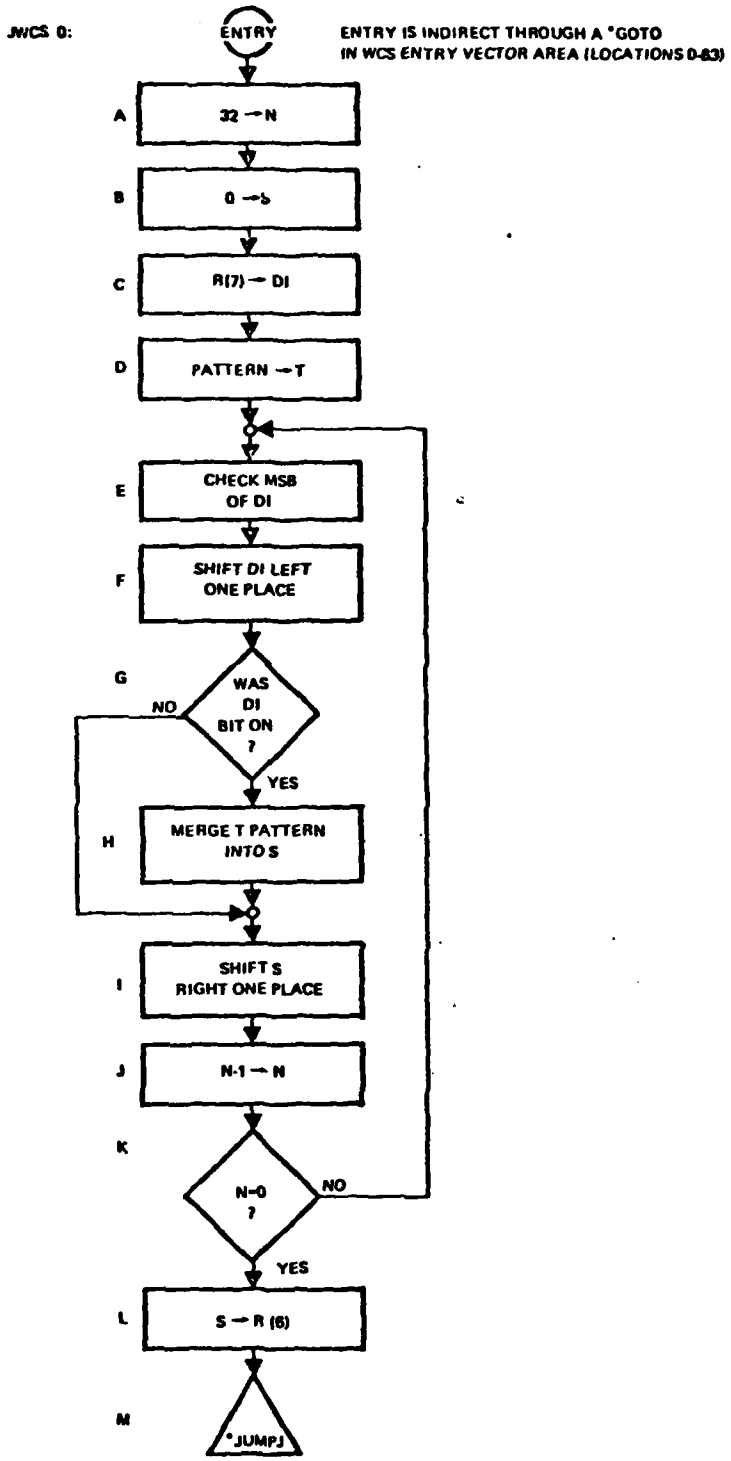


Figure 10-1. Sample Microprogram 1 Flowchart

```
$JOB MICRORUN
$ASSIGN1 BO=CC1
$EXECUTE MICRO
         $USE DEF.75F;
         *GOTO BITINV;
(@100)
*        MICROROUTINE WHOSE FUNCTION IS TO DO A BIT BY BIT SWAP
*        OF THE CONTENTS OF R7.
*        EXAMPLE, THE CONTENTS OF BIT 0 ARE MOVED TO BIT 31,
*                 THE CONTENTS OF BIT 1 ARE MOVED TO BIT 30, ETC
*        NOTE:   THIS PROGRAM HAS A BUG!
BITINV   DI=R(7);
         NU=@20000000              ;COUNT OFF 32 BITS FOR SHIFT
         T=@80000000,CLRS          ;SET NEW BIT INTO T AND CLRS S
BT2BT2   NOD=DI,DECRN,SHIFTDI(SLL) ;
         IF BMUXOO *GOTO $+2       ;CHECK FOR BIT ON
         S=S:T     ;BIT WAS SET MERGE A NEW ONE INTO THE RESULT
         SHIFTS(SRL),IF ZNCTRZ *GOTO BT2BT2     ;
         R(6)=S,*JUMPJ             ;DONE--RETURN SHIFTED BITS TO R6 AND
         SHIFTDI(SLL)                   ;
         $END
$ASSIGN1 BI=CC1
$EXECUTE MICROLD1
$OPTION 2 5
$EXECUTE ASSEMBLE
         PROGRAM    TESTWCS
*        PROGRAM TO TEST WCS ROUTINE WHICH IS TO INVERT THE
*        BITS IN R7.
INPDTA   DATAW      X'0F5A6670'     INPUT PATTERN
OUPDTA   DATAW      0    OUTPUT PATTERN
START    BOUND      1W
         LW         7,INPDTA
         TRR        7,4
*        JWCS       0
         NOP        TO FIT ONTO PROPER BOUNDARY
         GEN        16/X'FA00',16/X'0000'     FAKE WCS INSTRUCTION
         STW        6,OUPDTA        SAVE RESULT OF WCS ROUTINE
         LW         5,=C'JWCS'
         CALM       X'57'           ABORT TO FORCE CORE DUMP
         END        START
$EXECUTE GO
$EOJ
$$
```

Figure 10-2. First Generation of Bit Swap WCS Routine

```
$JOB MICRORUN2
$ASSIGN1 BO=CC1
$EXECUTE MICRO
         $USE DEF.75F;
         *GOTO BITINV;
(@100)
*        MICROROUTINE WHOSE FUNCTION IS TO DO A BIT BY BIT SWAP
*        OF THE CONTENTS OF R7.
*        EXAMPLE, THE CONTENTS OF BIT 0 ARE MOVED TO BIT 31,
*                 THE CONTENTS OF BIT 1 ARE MOVED TO BIT 30, ETC
*        THIS IS A FIXED VERSION OF THE EARLIER VERSION
*        THE EARLIER VERSION OVER SHIFTED THE OUTPUT
*        BECAUSE OF THE SHIFTS(SRL)
*        THE NEW VERSION BRINGS THE BIT INTO THE BOTTOM
*        AND THEN SHIFTS IT TO BIT 00-- THEREFORE WE
*        DON'T OVERSHIFT AND VERY IMPORTANTLY
*        WE DON'T LOSE ANY OF THE BITS
BITINV   DI=R(7);
         NU=@20000000              ;COUNT OFF 32 BITS FOR SHIFT
         T=@00000001,CLRS          ;SET NEW BIT INTO T AND CLR S
BT2BT2   NOD=DI,DECRN,SHIFTDI(SLL) ;
         IF BMUX00 *GOTO $+2       ;CHECK FOR BIT ON
         S=S:T     ;BIT WAS SET MERGE A NEW ONE INTO THE RESULT
         SHIFTS(SRC),IF %NCTRZ *GOTO BT2BT2     ;
         R(6)=S,*JUMPJ             ;DONE--RETURN SHIFTED BITS TO R6 AND
         $END
$ASSIGN1 BI=CC1
$EXECUTE MICROLD1
$OPTION 2 5
$EXECUTE ASSEMBLE
         PROGRAM    TESTWCS
*        PROGRAM TO TEST WCS ROUTINE WHICH IS TO INVERT THE
*        BITS IN R7.
INPDTA   DATAW      X'0F5A6670'     INPUT PATTERN
OUPDTA   DATAW      0     OUTPUT PATTERN
START    BOUND      1W
         LW         7,INPDTA
         TRR        7,4
*        JWCS       0
         NOP        TO FIT ONTO PROPER BOUNDARY
         GEN        16/X'FA00',16/X'0000'     FAKE WCS INSTRUCTION
         STW         6,OUPDTA       SAVE RESULT OF WCS ROUTINE
         LW          5,=C'JWCS'
         CALM        X'57'          ABORT TO FORCE CORE DUMP
         END         START
$OPTION DUMP
$EXECUTE GO
$EOJ
$$
```

Figure 10-3. Final Version of Bit Swap WCS Routine

*Logical (S-register bit 00 gets a Zero fill) and at the same time checking whether the N-Counter has been decremented to Zero as a result of having gone through the loop 32 times. If not done, a backward HOP is accomplished and the process is repeated.

If done, the S-register results are transferred to R(6) and then control is given back to the CROM driven Microengine.

When first run with the listed test programs and the binary input of:

00001111010110100110011001110000

the result was:

00000111001100110010110101111000

instead of the desired:

00001110011001100101101011110000

This indicated that the output has been overshifted. The first reaction is to check the count loop for the possibility of executing 33 times instead of 32. Careful desk checking reveals no logic or timing problems.

```
NU=Count

L1

L2

DECRN

L3

L4

IF NCTRZ
```

will cause the execution of the paths L1, L2, L3, L4, exactly Count number of times.

A better logical analysis of the S-register data flow reveals that there is only a 31 shift difference between bit positions 00 and 31 of the S-register.

Therefore, shifting the S-register 32 bits will result in the initial bit 00 value being shifted out of bit position 31.

Obviously, this is the explanation for the data patterns seen.

The first cure thought of may be finding a way to get a 33-bit wide register. This is impractical, however, since its logical mate, the DI register, is being shifted in the opposite direction because of the algorithm requirements.

It is apparent that a 32-bit register, when shifted circular, can seem to be a 33-bit register with only the last 32 bits saved. Therefore, line 18 is changed from a SHIFTS (SRL) to a SHIFTS (SRC), and the placement of the generation bit in the T-register is changed from bit 00 to bit 31, because when shifted Circular, bit 31 is also bit 01. The code at line 14 is changed from T=@80000000 to T=@00000001.

The reassembly and retest for an input of:

00001111010110100110011001110000

yielded the following:

00001110011001100101101011110000

CHECKLIST

This problem covered the following Microprogramming applications:

1. Literal Generation
2. Count Control
3. Shift Techniques
4. Register Assignment
5. Debugging Analysis

SAMPLE PROGRAM 2

The following program (Figure 10-4) takes an integer number in R(7) and converts it into a Floating-Point format equivalent in R(7). This routine is fully commented, therefore no textual support is presented.

SAMPLE PROGRAMS 3 AND 4

The following two programs (Figures 10-5 and 10-6) are presented to illustrate programming formats and the use of WCS to Read and/or Write to Main Memory from WCS. Both programs are adequately commented, so no further textual support is given.

```
$JOBMICRO.IR
$ALLOCATE 30000
$EXECUTE MICRO
        $USE DEF.75F;              ESTABLISH LANGUAGE DEFINITION SET
(@000)  *GOTO WCSC.RI;             VECTOR TO CONVERSION ROUTINE
(@100)
*    MICROROUTINE THAT WILL TAKE AN INTEGER NUMBER IN R(7) AND
*    CONVERT IT INTO A FLOATING POINT FORMAT EQUIVALENT IN R(7).
*
WCSC.RI  T=R(7),SAVESIGN;              GET INPUT AND HOLD SIGN
         T=0+T,ABST;                   USE INTEGER ABSOLUTE VALUE ONLY
         NU=@46000000,IF ALUZ *JUMPJ; INITIALIZE EXPONENT FOR BASE POINT
*     OF INTEGER, BUT IF INITIAL INTEGER ZERO, RETURN ZERO IMMEDIATELY
*
LARGE    NOD=@FF000000&T;              TEST FOR LARGE NUMBER
         *NOP;                         DELAY ONE CYCLE FOR TEST
         IF ALUZ *GOTO SMALL;          BRANCH IF NOT LARGE NUMBER
         TNIBR,INCRN,*HOP LARGE;       DIVIDE BY 16, BUMP EXPONENT, RETRY
*
SMALL    NOD=@00F00000&T;              CHECK FOR NORMALIZED FORM
         FULLMAR=T;                    MERGE FRACTION & EXPONENT
         S=0-MAR,IF NALUZ *GOTO CHEKSIGN; BRANCH IF NORMALIZED FORM
*  BUT FIRST SET UP TWO'S COMPLEMENT IN CASE INTEGER WAS NEGATIVE
          TNIBL,DECRN,*HOP SMALL;   INTEGER * 16, EXPONENT-1, RETRY
*
CHECKSIGN R(7)=MAR, IF %SIGNSAVE *JUMPJ; ALL FINISHED IF INTEGER POSITIVE
          R(7)=S,*JUMPJ;           TWO'S COMPLEMENT IF INTEGER NEGATIVE
          $END
$EXECUTE MICROLD1
$OPTION 2 5
$EXECUTE ASSEMBLE
         PROGRAM TESTWCS
*        THIS TEST PROGRAM DRIVES THE WCS ROUTINE FOR C.IR
*        IT TESTS FOR INPUT =0,1,2,3,4,5,6,7,8,9,10,15,16,17, ETC.
BEGADDR  ACW      $
         DATAW    C'BEGINNING OF TEST DATA INPUT'
TDTA     DATAW    0,1,2,3,4,5,6,7,8,9,10,15,16,17,32,64,128,256,;
                  512,1024,2048,4096,8192,16384,32768,65536,;
                  131072,262144,524288,1048576,2097152,4194304,;
                  8388608,16777216,33554432,67108864,134217728,;
                  268435456,536870912,1073741824,2147483647
         DATAW    -1,-2,-4,-8,-15,-16,-17, -8388608,-1073741824
TDTACNT  EQU      $-TDTA                LENGTH OF THE TEST TABLE
         DATAW    C'BEGINNING OF TEST DATA OUTPUT'
TDTO     REZ      TDTACNT          SAVE SPACE FOR RESULTS
TESTCNT  DATAW    0
         DATAW    C'END OF TEST DATA AREAS'
ENDADDR  ACW      $
START    BOUND    1W
         ZMW      TESTCNT               COUNTER FOR CONTROL
INLOOP   LW       1,TESTCNT             WHERE WE ARE IN TEST TABLE
         LW       7,TDTA,1              GET ONE TEST ITEM
         JWCS     0                     GOTO CONVERSION ROUTINE IN WCS
         LW       1,TESTCNT
         STW      7,TDTO,1              STORE RESULTANT
         ABM      29,TESTCNT
         LW       1,TESTCNT
         CI       1,TDTACNT
         BLT      INLOOP                DO THE WHOLE LIST
         LW       6,BEGADDR
```

Figure 10-4. Integer to Floating-Point WCS Routine (Sheet 1 of 2)

```
         LW        7,ENDADDR
         ZR        5
         CALM      X'4F'              CALL FOR DUMP OF ONLY OUR AREA
         LW        5,=C'JWCS'
         CALM      X'57'              ABORT TO FORCE CORE DUMP
         END       START
$EXECUTE GO
$EOJ
$$
```

Figure 10-4. Integer to Floating-Point WCS Routine (Sheet 2 of 2)

```
$JOB MICREAD
$ALLOCATE 30000
$EXECUTE MICRO
         $USE DEF.75F;
         $FORM 1,0
*        THIS MICROROUTINE IS ENTERED WITH JWCS 2
(@002)
         *GOTO WCS.READ;
(@200)
*        THIS IS A MICROUTINE TO READ MEMORY FROM AN ADDRESS
*        SPECIFIED IN R1.
*        RAW MEMORY WORD IS LOADED INTO R4
WCS.READ MAR=R(1), READ,FRCWORD;FORCES LOGICAL MEM SPACE READ OF WORD
*        THE READ WILL NOT GET STARTED UNTIL THE NEXT MICROINSTRUCTION
*        CYCLE, SO SINCE WE HAVE NOTHING TO DO WE MUST JUST KILL TIME
         *NOP                    ;
         R(4)=DI   ;CONTROL WILL WAIT AT THIS POINT UNTIL MEMORY IS
*                          COMPLETED WITH THE READ
*                   THE DATA FROM MEMORY IS NOW IN R4
         *JUMPJ;   GO BACK TO MACRO LEVEL
         $END
$EXECUTE MICROLD1
$OPTION 2 5
$EXECUTE ASSEMBLE
         PROGRAM   TESTWCS
BEGADDR  ACW       $
         DATAW     C'BEGINNING OF TEST DATA INPUT'
MEMDATA  DATAW     C'ABCD'          DATA TO BE LOADED
         DATAW     C'BEGINNING OF TEST DATA OUTPUT'
ODATA    DATAW     0                OUTPUT SPACE
         DATAW     C'END OF TEST DATA AREAS'
ENDADDR  ACW       $
START    BOUND     1W
         LW        4,=C'XXXX'       FILLER TO SEE EFFECT OF WCS ROUTINE
         LEA       1,MEMDATA        ADDRESS OF DATA TO GO TO R1
         JWCS      2                TRANSFER CONTROL TO WCS ENTRY ADDRESS
         STW       4,ODATA
         LW        6,BEGADDR
         LW        7,ENDADDR
         ZR        5
         CALM      X'4F'              CALL FOR DUMP OF ONLY OUR AREA
         LW        5,=C'JWCS'
         CALM      X'57'              ABORT TO FORCE CORE DUMP
         END       START
$EXECUTE GO
$EOJ
$$
```

Figure 10-5. Memory Read from WCS Routine

```
$JOB MICWRITE
$ALLOCATE 30000
$EXECUTE MICRO
         $USE DEF.75F;
         $FORM 1,0
*        THIS MICROROUTINE IS ENTERED WITH JWCS 3
(@003)
         *GOTO WCS.WRITE;
(@300)
*        THIS IS A MICROROUTINE TO WRITE MEMORY FROM AN ADDRESS
*        SPECIFIED IN R1.
*        DATA TO BE WRITTEN MUST BE PREVIOUSLY LOADED INTO R4
WCS.WRITE  MAR=R(1)                ;GET ADDRESS OF DATA WRITE
         T=R(4),WRITE,FRCWORD      ;AND PLUG THE DATA INTO T
*                                  AT THE SAME TIME THAT WE FORCE THE WR
*                                  WRITE OF THE FULLWORD
         *NOP                      ;
*                                  THIS NOP WILL GIVE T TIME TO GET ONTO
*                                  THE BUS--WE MAY CODE ANY OTHER MICRO
*                                  INSTRUCTION HERE THAT DOES NOT LOAD T
*                                  OR THE REGISTER FILE
*                                  ANY SUCH ATTEMPTED LOADING
*                                  WILL FORCE A CONTROL STRUCTURE WAIT
*                                  UNTIL THE SELBUS HAS COMPLETED
*                                  TAKING THE DATA FROM T.
         *JUMPJ:    GO BACK TO MACRO LEVEL
         $END
$EXECUTE MICROLD1
$OPTION 2 5
$EXECUTE ASSEMBLE
         PROGRAM   TESTWCS
BEGADDR  ACW       $
         DATAW     C'BEGINNING OF TEST DATA INPUT'
MEMDATA  DATAW     C'ABCD'          DATA TO BE OVERWRITTEN
         DATAW     C'BEGINNING OF TEST DATA OUTPUT'
         DATAW     C'END OF TEST DATA AREAS'
ENDADDR  ACW       $
START    BOUND     1W
         LW        4,=C'XXXX'      FILLER TO SEE EFFECT OF WCS ROUTINE
         LEA       1,MEMDATA       ADDRESS OF DATA TO GO TO R1
         JWCS      3               TRANSFER CONTROL TO WCS ENTRY ADDRESS
         LW        6,BEGADDR
         LW        7,ENDADDR
         ZR        5
         CALM      X'4F'           CALL FOR DUMP OF ONLY OUR AREA
         LW        5,C='JWCS'
         CALM      X'57'           ABORT TO FORCE CORE DUMP
         END       START
$EXECUTE GO
$EOJ
$$
```

Figure 10-6. Memory Write from WCS Routine


END
DATE
FILMED
1-81
DTIC

AD-A106 779 FLORIDA INST OF TECH MELBOURNE DEPT OF ELECTRICAL AN--ETC F/G 14/2
IMPLEMENTATION OF THE RECOMMENDATIONS MADE ON THE TECHNICAL REP--ETC(U)
JUN 81 J HADJILOGIOU AFOSR-80-0120

UNCLASSIFIED AFOSR-TR-81-0704 NL

SUPPLEMENTARY

INFORMATION

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Correction 22 Sep 82

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFOSR-TR- 81-0704
2. GOVT ACCESSION NO: AD-A106779
3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): IMPLEMENTATION OF THE RECOMMENDATIONS MADE ON THE TECHNICAL REPORT TITLED "ANAYSIS OF ADVANCED SIMULATOR FOR PILOT TRAINING"
5. TYPE OF REPORT & PERIOD COVERED: Final Report
6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): John Hadjilogiou
Should be AFOSR-80-0120
8. CONTRACT OR GRANT NUMBER(s): AFOSR-80-0120

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Florida Institute of Technology
Dept of Electrical & Computer Engineering
Melbourne, Florida 32901
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F 2313/D9

11. CONTROLLING OFFICE NAME AND ADDRESS:
Air Force Office of Scientific Resch/NL
Bolling AFB, DC 20332
12. REPORT DATE: June 1981
13. NUMBER OF PAGES: 109

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office):
15. SECURITY CLASS. (of this report): Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microprogrammable processor, control logic for 32175 computer mocro coding of ASPT simulator.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This project resulted in a report detailing specific quidelines for writing and testing custom micro-programs for the 32/75 computer. The micro-program instruction format is analyed in detail and then illustrated by a concrete example.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)